ZX COMPUTING MONTHLY

AN ARGUS SPECIALIST PUBLICATION JANUARY 1987 £1.50

FOR ALL SINCLAIR USERS



TUNE COMPOSER FOR SPECTRUM AND QL

CREATE YOUR OWN . . .

Adventure Landscapes

Illuminated Text

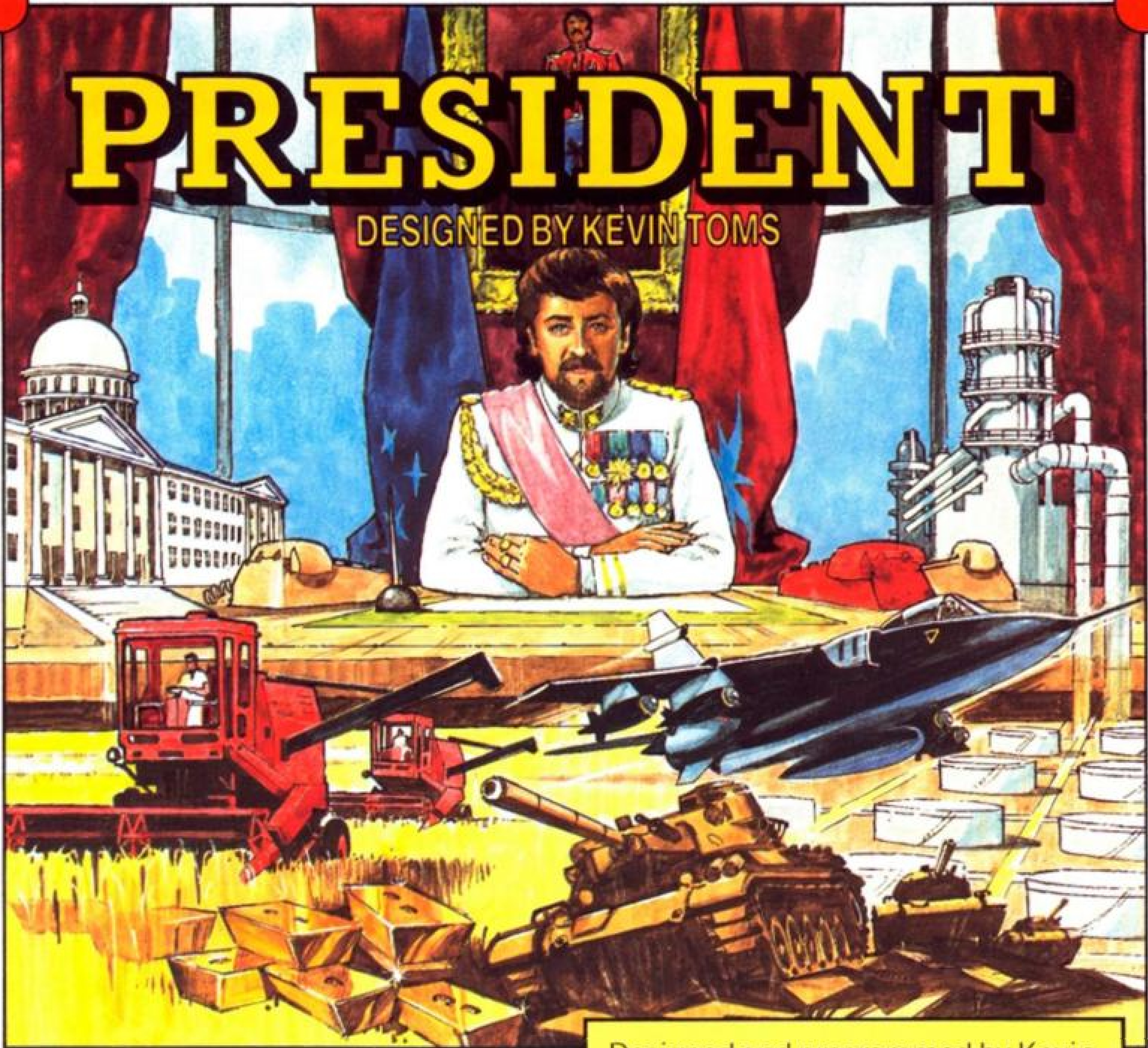
Technical Graphics

SPECWORD:

Type-in word processor for 48/128 Spectrum

MONSTER HITS! ■ Thanatos ■ Infiltrator ■ Avenger ■

WIN! ■ Rogue Trooper ■ Marble Madness ■ Cop-Out ■



Throughout history, in times of anguish and need, there have arisen great leaders who have turned away the tides of defeat, deprivation and slavery. They have secured not only protection and prosperity for their peoples but also a place in the history of mankind.

Dare you stand beside them?

Measure yourself against the evil forces of today – total war – epidemic – starvation – poverty – and prove that you have the power to be

PRESIDENT

AMSTRAD £9.95 SPECTRUM £8.95

Designed and programmed by Kevin Toms, the creator of the all-time classic "FOOTBALL MANAGER"

FEATURING:

- Oil Exploration and Development
- Battle Tactics and Defence Strategy
- Politics – including Food Policy, Health, Economic Problems, Vote Winning Ploys etc.
- Finance – Oil Sales, Gold Reserves, Exchange Rates...and of course – Elections

Now available on Amstrad and Spectrum.

Addictive

Addictive Games Ltd
10 Albert Road · Bournemouth
Dorset BH1 1BZ · Tel: 0202-296404

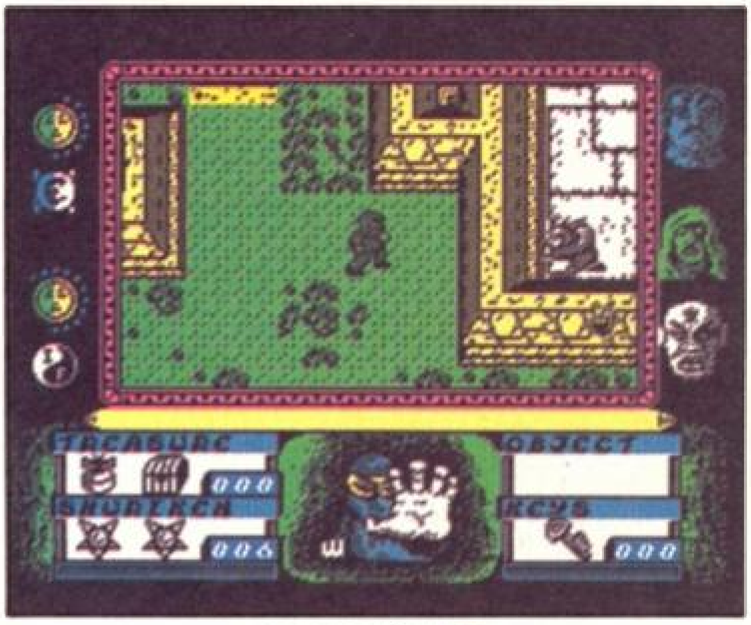
Post to: Addictive Games · 10 Albert Road · Bournemouth · Dorset BH1 1BZ

☐ President · Amstrad .... £9·95 each
☐ President · Spectrum .... £8·95 each

Cheque/Postal Order/Access ....
Name ....
Address ....
....
.... Tel. ....

CONTENTS

FEATURES



Avenger review (82)

PROGRAMMING PROJECTS

REGULARS

REVIEWS



Thanatos takes to the air (46)

COMPETITIONS

Front Cover Piano: Harrods of Knightsbridge

N·E·W·S

The Sound of Cheetah



Cheetah have added a MIDI keyboard to their ever-expanding range of musical peripherals. Called the MK5 it is a full-size, five octave polyphonic keyboard with 61 keys and pitch bend wheel. Up to 128 patches can be recalled from the chosen sound module. Other features include a hold function, octave shift, LED display, MIDI out (assignable to any one of 16 MINI channels) and it comes complete with a metal case.

The keyboard can be utilised to a computer music controller via the Cheetah MIDI interface (£29.95). Compatible with the 128/+2, the keyboard will enable the user to edit and play a whole variety of computer-generated sounds.

Cheetah plan further additions to their "slave keyboard" in the future to create even more possibilities for the system.

The MK5 retails at £99.95.

Cheetah can be contacted on (0222) 777337.

Readership Survey

It's time to look in the mirror with some findings from the ZX readers' survey.

The sections that you consider best covered in ZX are Short Cuts, Utility Listings and Programming Series. These categories also figure highly in your requests for further coverage.

Not surprisingly the machine owned by most ZX readers is the Spectrum or Spectrum+ but there is now a sizeable contingent of 128 owners and 36% of Spectrum owners said they intend to upgrade to the 128.

As for machine use, most ZX readers spend their time involved in original programming closely followed by games playing and typing in utility listings. Among the game players there was an almost even split between adventurers, arcade fanatics and strategists.

By far the most common piece of hardware owned was a printer with 72 per cent of readers with one in their possession.

Thank you once again for replying to our readers' survey and for ten correspondents there is a free ZX subscription, they are: J. E. Daniels, Milton Keynes, J. A. Sitton, Stevenage, M. Bedford Whit, Birmingham, Peter Booth, Merseyside, Craig Jones, Kidderminster, J. S. Dare, Kempston, T. F. Barber, Glasgow, T. Turner, Hull, C. J. Morrison, Edinburgh, Ernest Taylor, Cleveland.

Trivial Pursuit Winners

The ZX Trivial Pursuit champion is Robert Burgess of Rotherham who will be representing ZX in the final of Domarks Golden Trivia challenge. Robert will be competing against Trivial Pursuit champions from other computer magazines for a grand prize of a gold Trivial Pursuit set.

In addition, Robert wins a copy of Domarks Trivial Pursuit as do four runners-up: R. Sands, Skipton, Jill Edmunds, Bristol, J. Riddell, Herne Bay and M. Scott from Gateshead.

Seven other entrants win Trivial Pursuit After Dinner Mints. They are: C. J. Geggus, Romford, A. Welsh, Glasgow, A. Beale, Blandford Forum, A. Bradbury, Redditch, S. Burr, Aylesbury, S. Hollinshead, St Ives and Mathew Holder, Stanmore.

Destruction Test

Konix are planning to test the durability of its Speed King joystick by attaching it to a specially designed machine that will manipulate it at high speed, non-stop, for as long as the joystick will last.

For those familiar with Daley Thompson's decathlon, the constant rate of play will be equivalent to doing the 100 metres indefinitely.

Konix are inviting guesses from one and all on the lifespan of the joystick under these conditions and are offering a £100 prize to the person whose guess is closest. The company concede that they have no idea whether it will be days, weeks or months before the joystick seizes up.

Those tempted to take a guess should submit their entries on a postcard giving their name, address and estimate to Daley Thompson's Decathlon Test, Konix, c/o Solution Public Relations, 2 Wellingtonia Court, Varndean Park, Brighton BN1 6TD. The closing date for entries is 31st January, 1987.

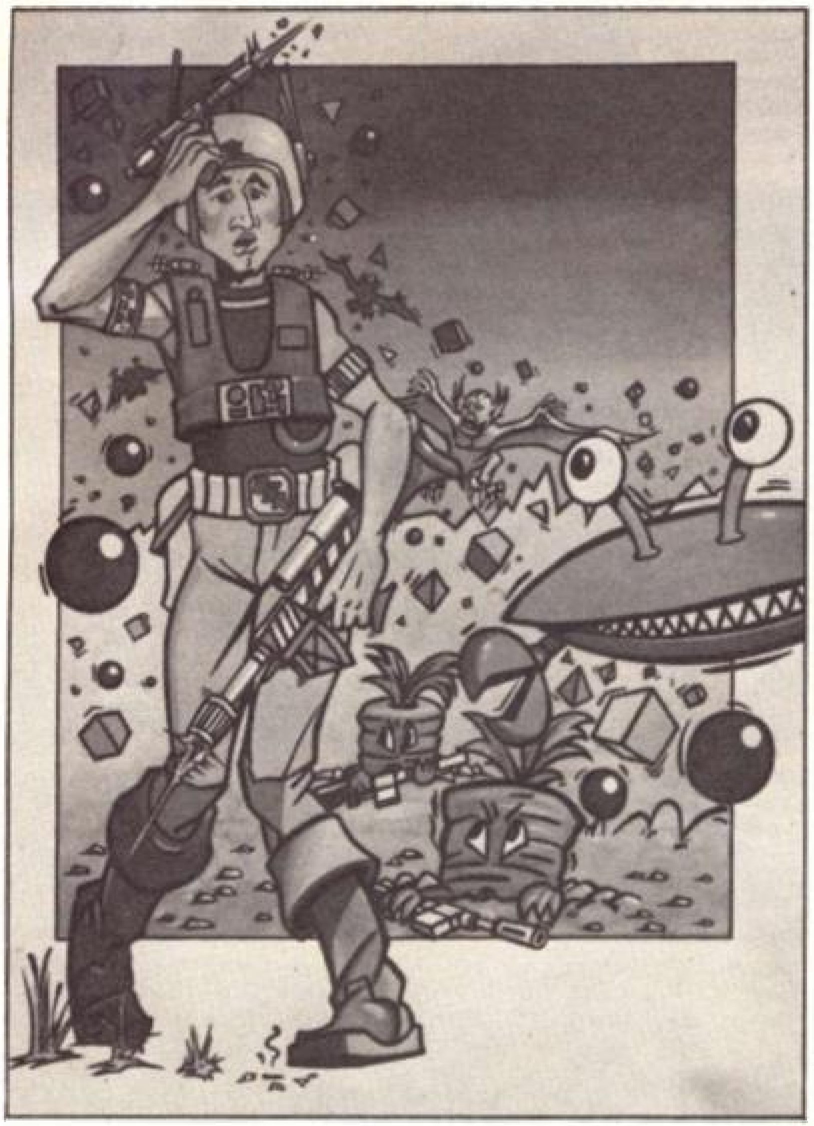
Editor: **Bryan Ralph**
Assistant Editor: **Cliff Joseph**
Consultant Editor: **Ray Elder**
Advertising Manager: **John McGarry**

Design: **Argus Design**
A.S.P. Advertising and Editorial
No. 1 Golden Square, London W1R 3AB 01-437-0626

Printed by Chase Web, Estover, Plymouth.
Advertisement Copy Controler: Andy Selwood
Distributed by: Argus Press Sales and Distribution Ltd, 12-18 Paul Street, London EC2A 4JS

ZX Computing Monthly is published on the fourth Friday of each month. Subscription rates cn be obtained from ZX Subscriptions, Infonet, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP51 1BB.

An Absorbing Shock



Martechs latest character creation is an upwardly mobile Super hero vying to get into the big league with the likes of Batman and Superman. But it's hard making your way in the annals of modern legend when you have to tackle enemies such as armies of bouncing balls and carrots carrying machine guns. Cosmic Shock Absorber is the name of the apprentice Super hero in a 3D shoot-em-up (or root-em-up in the case of the carrots) that costs £7.95.

Apologies

A couple of gremlins crept into the works in last month's issue. Druid from Firebird was left bereft of a monster rating — it was in fact a GREAT.

Also, Tarzan, the computer game is not published by CRL as stated — it is published by Martech.

Deeply Struck

Deep Strike is Durells latest release, a WW1 aerial combat game fought out over enemy territory. As the pilot of a bi-plane you must protect a bomber as its assaulted from all sides by ack-ack fire, enemy fighters as well as see it safely round groups of barrage balloons and hill formations. Deep Strike retails at £9.95.

Well Brian . . .

Although he hasn't said it's "a game of two halfs" (yet) Brian Clough, the touch line philosopher, is lending his name to a new computer/board game from CDS. The package, called Brian Clough's Football Fortunes consists of cassette, playing board, counters, football player cards, immunity cards and vast sums of money for transfer deals. It's a team manager game for up to five players and the object is to take your team to the top of the league, as well as carry off the FA Cup. Transferring the game to your home costs a four-figure fee — £14.95.

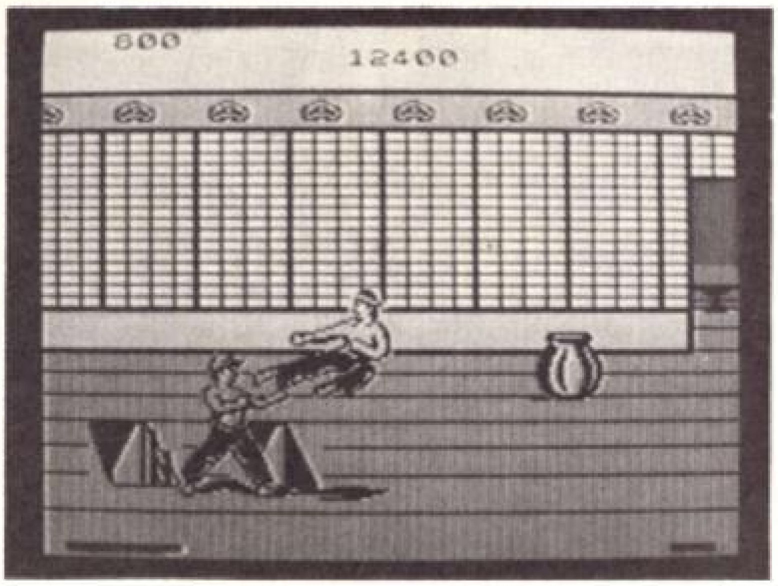
Staying in the football vein, Argus Press Software's Grandslam label is releasing Peter Shilton's Handball Maradona! A 3D goalkeeping simulation in which you must keep a clean sheet in a variety of game situations. Handball Maradona costs £6.95.

The House of Combat

Melbourne House, as well as releasing Fist II, the follow-up to Way of the Exploding Fist, have a number of other combat-based action games in the pipeline. Knucklebusters is set in an urban nightmare future where criminals are transformed into android slaves. One rebels and the aim of the game is to guide him through the six zones of the city through hosts of kamikaze robots to freedom.

Judge Dredd, the 2000AD comic character will be fighting the forces of chaos in Mega City 1. The futuristic law enforcer has to contend with Robodogs, rats and anarchists but when the long arm of the law has a gun that can fire heat-seeking bullets, the scales of justice may be tipped his way.

Kwah! the sequel to Redhawk will be continuing the scrolling cartoon format of the original in an adventure which pits Redhawk against the sinister scientist Dr Lee. Knuckledusters and Judge Dredd will retail at £8.95, Redhawk costs £7.95.



Fist II.

Spectrum Games Top Ten

1	(3)	Computer Hits 3	Beau Jolly
2	(7)	Trivial Pursuit	Domark
3	(4)	Paper Boy	Elite
4	(1)	Infiltrator	U.S. Gold
5	( )	Avenger	Gremlin
6	( )	The Great Escape	Ocean
7	( )	Dandy	Electric Dreams
8	( )	Five Star Games	Beau Jolly
9	( )	Trailblazer	Gremlin
10	( )	Tarzan	Martech

Chart supplied by W.H. Smith

Trivial Pursuit continues to bounce around the charts. A former number one, it dropped back to number seven last month, but is yet again challenging for the top spot. Following up their success, Domark have now released a Young Players' edition, £14.95, or if you already own the Genus edition the Young Players' question pack costs £7.95.

Jewels of Darkness Winners

The first ten entries pulled out of the hat in our "spot the similarities" contest featuring our Level Nine front cover and previous L9 game covers, each win a copy of Rainbird's Jewels of Darkness.

They are: Samer Kurdi, Amman, Mrs A. Trotman, Lancing, Duncan North, Sheffield, S. J. Madeley, Middlewich, Cecil Westerchoff, Utrecht, Mathew Austin, Chesham, A. Hulmes, Altrincham, K. E. Rankin, Aldridge, Tommy Tay, Oberusel, Anthony Ross, Montana, USA.

Up in the Clouds

Ian Martin, programmer of Ace, has come up with a "flying motor bike simulation" as part of his new game for Cascade, called Sky Runner.

"The idea for Sky Runner came from a couple of my favourite movies," said Ian, "Dune and Return of the Jedi. The scene in Jedi where the flying motor cycles are hurtling through the trees was by far the most exciting sequence. That was the inspiration for the action scenes in Sky Runner.

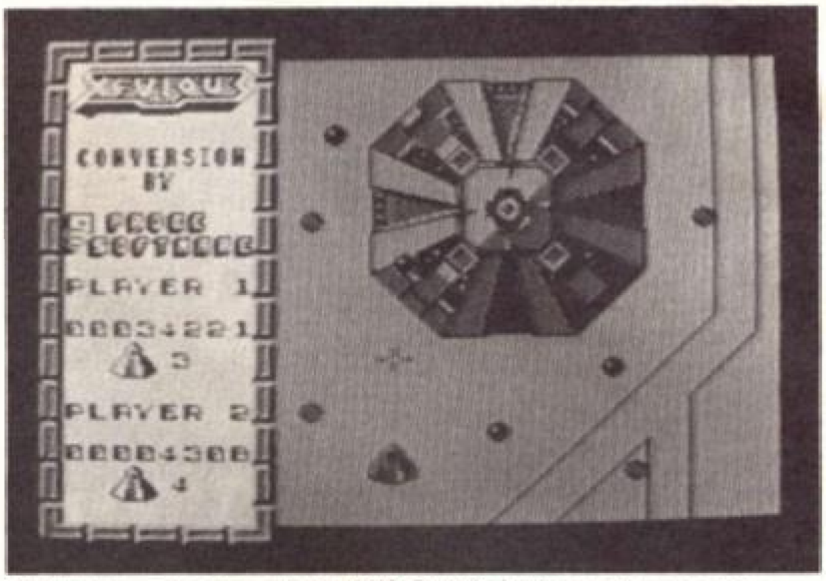
"The story line for the game could be considered controversial in that it is based on drug busting on an inter-planetary level. Future society is subdued into docile obedience by the drug SKY and it is your function to terminate the drug harvesting operation.

Sky Runner is priced at £9.95.



Americana Winners

Ten ZX readers carry off a quintet of Americana titles from U.S. Gold. They are: M. A. Dalli, London SW8, Scott Jones, Heston, Melanie Hughes, Alderley Edge, Ian Macauley, Dudley, Allan Gibbs, Ipswich, Mark Syder, Windle St. Helens, Jean Pierre Clegg, London W1, Phillip Goodyear, Market Rasen, David Brewster, Kircaldy, S. Turner, Isle of Wight.



***Xevious** — A screenshot of US Gold's forthcoming arcade conversion in which your objective is to destroy a vast mothership and stem an alien invasion.*

New Keyboard for QL

Schon Keyboards, a new company, are marketing a replacement keyboard for the QL. Comprising of two units (the keyboard and replacement housing) it costs £54.95.

The keyboard has 64 black, full travel "two shot" moulded keys and a full-sized spacebar. The five function keys are distinguished by being coloured red.

As the new housing is similar to the original, all peripherals will remain compatible, although new power and microdrive LED's have been added.

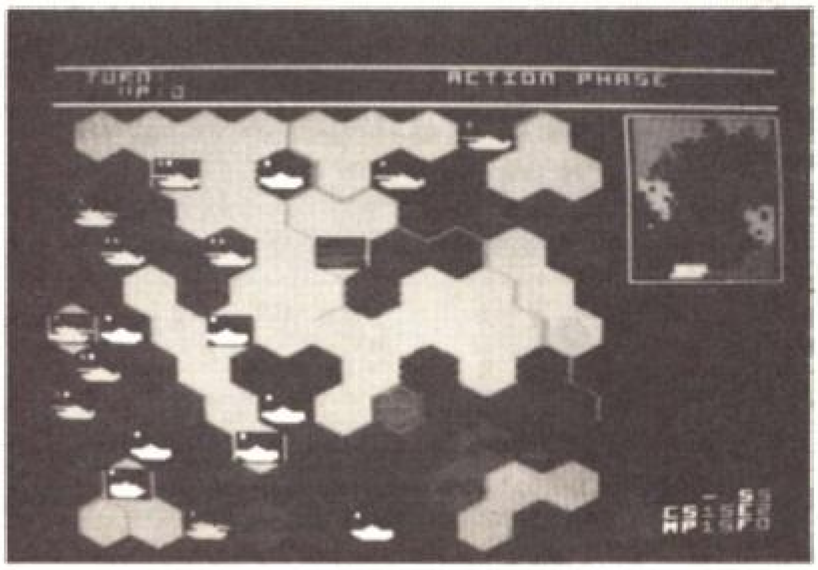
The fitting time for the new keyboard is estimated at only five minutes.

Schon Keyboards: 04865 3836.

Wargames

PSS have announced the latest releases in their Wargamers series. Set in the time of the Roman Empire, Annals of Rome puts you in charge of the ruling power group in the senate and your task is to further the expansion of the Empire's territories.

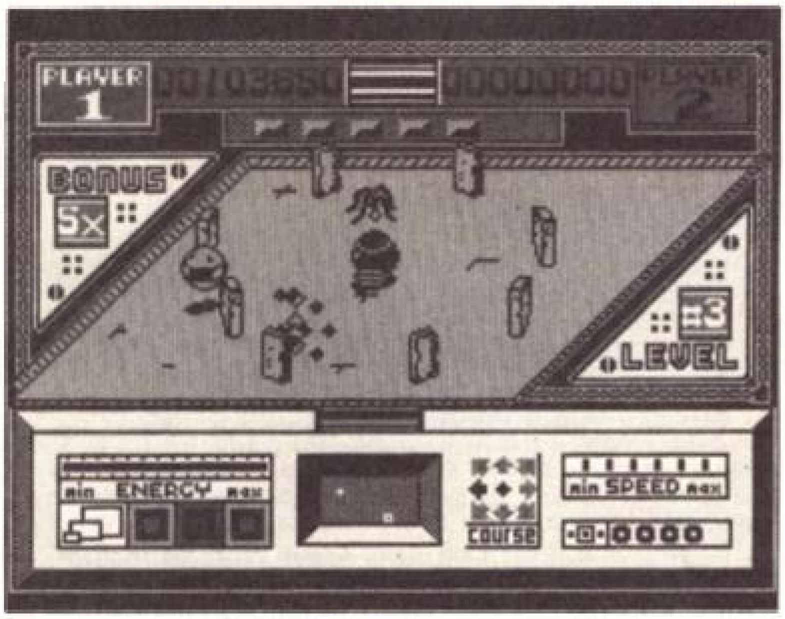
With Battlefield Germany we are pitched into contemporary times with a confrontation based on a hypothetical offensive into Western Germany by Warsaw Pact forces. Annals of Rome and Battlefield Germany both cost £12.95.



Battlefield Germany

Streetwise

The first two releases on Domark's new arcade label, Streetwise, are Orbix the Terrorball and Kat Trap. Described as a "bouncing shoot-em-up" Orbix the Terrorball is a spherical space ship under your command in a mission to rescue a space craft stranded on a hostile planet. Kat Trap puts you in control of a mechanical cat-like robot in a multi-levelled blast the nasties type game. Orbix costs £7.95 and Kat Trap retails at £8.95.



Orbix the Terrorball.

NEWS



SpecDrum
Digital Drum system for the Spectrum 48K/128K/+2
£29.95
Sweet Talker
Compatible with Interface I & II and Spectrum 16K, 48K, 128K, +2.
Complete with cassette and full instructions.
£24.95
Mega-Sound
For 48K Spectrum and ZX Spectrum+. Achieve amazing sound capabilities that your Spectrum has been lacking.
£10.95
Spectrum Joystick Interface
For Spectrum 48K/128K/+2. Accepts any Atari style joystick including Cheetah 125+, Mach 1+. For Kempston Compatability
£9.75
Sound Sampler
17·5KHz Bandwidth
Compatible with Spectrum 48K, 128K, +2.
£44.95
Midi Interface
Compatible with Spectrum 48K, 128K, +2.
£49.95
Cassette/Data Recorder
High quality cassette/data recorders for your home computer will allow easy loading of even the most stubborn programs.
£24.95
MK5 Midi Keyboard
A full size 5 octave polyphonic keyboard containing a sophisticated and very powerful computer, suitable for a wide range of uses
£99.95
Peripheral Power
Mach I+ Joystick
Probably the best Joystick in the world.
• Auto Fire
• Metal Shaft
• +2 Selector Connector
• 12 Month warranty.
£14.95
125+ Joystick
• 4 extremely sensitive fire buttons
• Auto fire switch
• +2 Selector connector
• Uniquely styled hand grip
• 12 Month warranty.
£8.95
R.A.T. Remote Action Transmitter
• Infra Red transmission
• Touch control
• No extra software required
• Compatible with all Sinclair/Cheetah peripherals via the rear edge connector.
£19.95
+2 Selector Connector
Sinclair Joystick Adaptor for connecting ZX Spectrum +2 computers to most joysticks including Cheetah 125 and Mach I
£2.99
56 Way Extension Connector
Cheetah's 6" long extension cable enables Spectrum peripherals to be distanced from your computer.
£7.95
Aerial Splitter
Cheetah's neat splitter unit complete with self-adhesive pad allows you to keep your T.V. and computer aerial leads plugged in without disturbing the picture.
£2.25
Spectrum 128/+2 to Midi Lead
Midi control lead for Spectrum 128K/+2 computer.
£9.75
Split Extension Connector
Enables 2 Spectrum peripherals to be connected to the computer bus.
£10.99
Prices include VAT, postage & packing.
Delivery normally 14 days.
Export orders at no extra cost.
Dealer enquiries welcome.
Cheetah, products available from branches of
Dixons WHSMITH
High St. Stores and all good computer shops.
Cheetah Marketing
CHEETAH MARKETING LTD.
1 WILLOWBROOK SCIENCE PARK
CRICKHOWELL ROAD, ST. MELLONS, CARDIFF
TEL: CARDIFF (0222) 777337 TELEX: 497455
FAX: 0222 779404

PROJECT ZX

Many of us enjoy programming but unless we have an objective in mind when we begin we often find that we sit at the computer and produce yet another variation of 'Mastermind' or some other program dinosaur and the resulting code is a typical example of the worst kind of spaghetti programming.

Not very satisfying, even if it works.

The aim of this occasional series is to present a problem, perhaps a complete program or maybe a specific routine for you to apply your talents to and to discuss the possible solutions, their suitability, alternatives and possible developments.

I hope we will generate an **interactive** section of the magazine where **you** will write to me and comment, criticise, offer solutions and even suggest possible problems for future analysis. ZXC has one of the largest and most dedicated user bases and we hope to offer you a means of expressing yourselves in a creative and exciting way.

Do not worry that you may be inexperienced and feel your ideas are not good enough, many's the time when I've looked at a piece of programming from a reader who states that they are "sorry that it may not be good enough as they're a newcomer" only to find it a sophisticated and ingenious bit of work!

Means to an end

I propose to tackle each problem in the following general outline, this is subject to modification as we are always open to suggestions, but to begin with I propose the following steps:

1. Identity of problem concisely.
2. Break the task into essential sections.
3. Discuss ways of implementing each of the sections.
4. Discuss possible developments.

Step 3 may also debate the suitability of structured

Ray Elder introduces a new series in which ZX readers can pit their wits against challenging programming problems

techniques and/or machine dependent techniques.

So without further ado let's look at the problem which inspired this article.

Code Puzzle

In a magazine that I infrequently purchase they ran a competition in which they presented us with the problem of decoding three messages which consisted of groups of letters into sensible, meaningful words or phrases. It was obvious that they were not anagrams and also had to be relatively simple so that most of us would have a chance to compete.

From this it seemed reasonable to me that they had used the technique of shifting letters along the alphabetic matrix by varying amounts, for example they might have used an offset of two to produce **Figure 1.**

Notice the 'wrap round' effect on Y and Z, now, as we all remember from our schooldays, to code the words required (eg. HELLO) you look up each letter and write down the corresponding one beneath it, so HELLO becomes FCJJM.

Given the task to decode such a message then the intuitive or logical part of a human often takes over, a three letter word is likely to be THE or AND, a single letter is likely to be I or A and the most frequently used letter is often E. However this depends on the message being long enough to be able to identify these clues and try them out, if the message is very brief then it is much harder and often the only way is to patiently try all 26 offsets until the words make sense.

An ideal job for the computer is to set up all the offsets and present the decoded words for the operator to scan to spot if they make sense, so:

1. Identify

The program needs to accept coded words input from the keyboard, perform each of the 26 offset letter comparisons in trun, presenting the results to the operator for their assessment as to validity.

2. Breakdown

a) INPUT words or phrase
b) PROCESS each letter by adding (or subtracting) a constant value to it
c) OUTPUT processed letters of the word or phrase to screen (or printer)
d) REPEAT sections b & c 26 times with value of constant increased by 1 each time (26 letters of alphabet)
e) END

3. Possible ways

Taking each section separately, first we have to decide the form in which we will type in the coded message, letter at a time, a word or complete phrase. Also depending on this will be the means of storing it, a single string for the whole thing? dimensioned array perhaps for single words or letters.

If we use an array then we need some input to find out the number of elements needed, on the other hand will it be more awkward examining a single string for the end of words?

Probably the most simple means of entry would be to enter the lot, provided it wasn't excessively long, in a single string with single word entry being useful for short phrases and single letter being somewhat more lengthy to do. Not forgetting that for both the

final options a prompt and input will be needed to establish either the number of letters or words to be entered.

The method I chose was to enter individual words into a preset array, however I knew that each phrase was only three words long. With hindsight and with the intention of making the program flexible I am leaning towards entering the whole phrase in one go as being the best option. This will depend on the method chosen to process the letters in the end.

One solution

In my opinion, remembering that it was short and simple, I set up numerical arrays for each word with a numberical variable for each word holding the LENgth of that word. These elements held the base number for each letter A=1, B=2 etc. obtained by finding the ASCII value of each letter and subtracting 64.

Next I set up a control string containing the alphabet, all, by the way in upper case. This last is another consideration which needs deciding on.

The actual processing was done by setting up a loop of 1 to 26 and shifting the control string 1 place left each pass — by LET c$=c$(2 TO)+c$(1) — and then printing out the characters in that new string that were obtained by the values of the word arrays. This used several FOR/NEXT loops and though it worked, it was very inelegant.

It would be easier to set up a numeric control variable which increased by one each time, it could even be the loop counter, and add it to the ASCII code of the letter, this could be restored in another variable or printed directly to the screen.

With this latter method you would have to check for spaces (if the phrase was entered as a complete string) and the wrap round effect: ie. any code over 90 would have to have 26 taken away from it.

It may be possible to use the fact that the address the computer uses to find the character set is given by system variable CHARS at 23606 and 23607, increasing 23606 by eight each time has the same effect as the last suggestion except it may be more difficult to allow for the wrap effect.

Finally the output, screen or printer? Should it wait for a keypress each time to a new set of possible decoded letters has been produced or just pause for a predetermined time? Should the screen be cleared or scrolled? Perhaps an option only to print selected attempts might be included.

All this will depend on the projected length of the code to be handled, and possibly on which of the methods of processing was chosen, and of course whether you own a printer!

I am aware that there may be other, better ways of going about the task, but this was intended just to throw out some ideas. I hope you'll throw some back at me.

4. Development

So we get to the solution of our easy code. Now, how about designing a system to encode messages in the first place, a relatively easy task.

Would it be worth a routine to check that the words haven't simply been reversed (as in Hewson's famous Seiddab aliens)?

It may be useful to include an anagram or scramble routine.

Could a program be developed to decode more complex codes?

Could the program identify a possible solution on its own without needing a human to sit and look at each offering? A Dictionary program such as a spelling checker would be useful for this. Is there one on the market? Could you write one? How would the words be stored (Microdrives at least!).

Over to you

I hope this gets the old grey matter working, I would be very interested in seeing your programs to perform the set task (not including any developments). My gash version was 23 lines long, if your program is equal to or shorter than this then pop it in the post, in this case we will accept handwritten or printout of your code without a cassette tape and the most elegant solution will be featured in a future issue — with a suitable reward of course!

Figure 1:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Y Z A B C D E F G H I J K L M N O P Q R S T U V W X

FAT WORM

blows a sparky

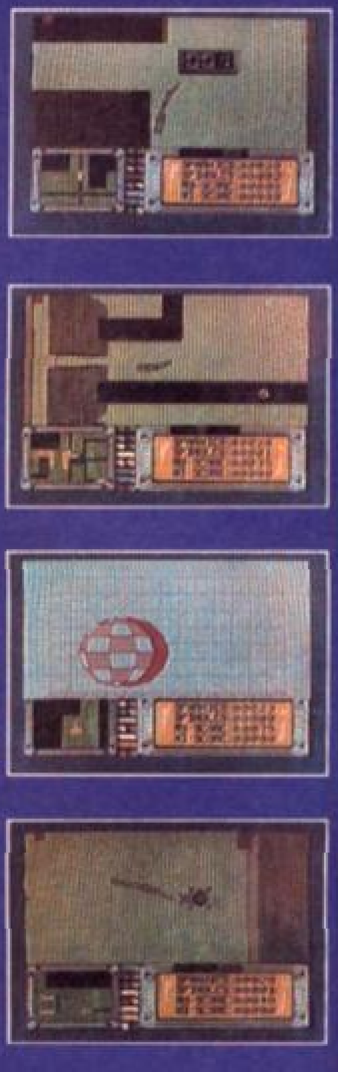
Now for something completely different. You are an insidious little WORMIE being chased through a micro-electronic labyrinth (you guessed it — a Sinclair Spectrum!) by CREEPERS in SPUTNIKS and CRAWLERS on foot (feets?). Defend yourself by shooting BURPER SPARKIES at the CRAWLERS, and BLASTER SPARKIES to take out the SPUTNIKS. You'll see the computer board in a smooth-scrolling 3D viewed from above (yawn, yawn — just another bit of mega-programming), as you crawl around in search of a DISK DRIVE on which to CLONE yourself. First you'll need to find fifty SPINDLES to eat, which will replenish your supply of SPARKIES to shoot at the BUGS. How long can you crawl down a DATA BUS? Find out how refreshing a DE-BUGGER feels when you're stuck in a Spectrum and covered in CRAWLIES. This game is like... WELL CRUCIAL MAN

Available **October 20th**

Trade enquiries to **Centresoft** on **021 356 3388**

Any Spectrum £9.95





DURELL

software getting harder....


DEEP STRIKE
R.R.P. £9.95
Spectrum & Amstrad
available November 20th
Commodore 64
available January 20th
Trade enquiries to Centresoft
on 021 356 3388
Sales dept.,
Castle Lodge, Castle Green,
Taunton, Somerset TA1 4AB
England
Telephone (0823) 54489 & 54029
DURELL
software getting harder . . . .

tune composer

A programming duet to convert your Spectrum or QL into a musical keyboard. Your conductor: David Nowotnik.

Last month's issue of ZXC contained my game program for Christmas, for both Spectrum and QL computers. To those who typed in the listing, I hope you not only enjoyed playing "Santa's Dilemma", but got some pleasure from hearing your computer play a Christmas carol.

That tune was programmed largely by trial and error; the BEEP command on both Spectrum and QL (particularly the latter) is difficult to control. So, as the hours ticked away while I tried to get the tunes right, I thought that there must be a better way. And the listings appearing in figs. 1 and 2 are the result.

Both programs, one for 48K Spectrum and Spectrum Plus **(fig. 1)** and QL **(fig. 2)** allow your computer keyboard to be used as a musical keyboard, stretching just over one octave. The two programs work in much the same way, so a single description of how to use them should suffice. So, once you have typed in the appropriate listing, SAVE and RUN it.

Typing in the listings should be straight forward, apart, perhaps, from the graphics characters of line 730 in the Spectrum listing. Here, once you have entered 'LET a$="', press CAPS, and keep it pressed down while you press the following sequence of keys:
4,9,5,9,3,9,5,9
then close the quotes, and press ENTER.

Figure 1

Tune Maker — Spectrum Listing

```
100 REM   *** Composer ***
110 REM    David Nowotnik
120 REM    October, 1986
130 REM
140 REM    INIT
150 REM
160 CLEAR 29999
170 LET start=30000
180 LET tune=0: LET add=start
190 PAPER 2: INK 7
200 REM
210 REM    MAIN LOOP
220 REM
230 CLS
240 PRINT  AT 2,10;"TUNE MAKER"
250 PRINT  AT 4,4;"Select:-"
260 PRINT  AT 8,4;"1. PRACTISE"
270 PRINT "    2. RECORD"
280 PRINT "    3. PLAYBACK"
290 PRINT "    4. SAVE"
300 PRINT "    5. LOAD"
310 PRINT "    6. EDIT"
320 PRINT "    7. EXIT"
330 LET z$= INKEY$
340 IF z$<"1" OR z$>"8" THEN  GO TO 330
350 LET sub= VAL z$
360 GO SUB sub*1000
370 IF sub=7 THEN  STOP
380 GO TO 210
390 REM
400 REM WHICH KEY
410 LET qq= PEEK 23556
420 LET key=254
430 IF qq=48 THEN  LET key=255
440 IF qq=65 THEN  LET key=4
450 IF qq=83 THEN  LET key=5
460 IF qq=69 THEN  LET key=6
470 IF qq=68 THEN  LET key=7
480 IF qq=82 THEN  LET key=8
490 IF qq=70 THEN  LET key=9
500 IF qq=84 THEN  LET key=10
510 IF qq=71 THEN  LET key=11
520 IF qq=72 THEN  LET key=12
530 IF qq=85 THEN  LET key=13
540 IF qq=74 THEN  LET key=14
550 IF qq=73 THEN  LET key=15
560 IF qq=75 THEN  LET key=16
570 IF qq=76 THEN  LET key=17
580 IF qq=80 THEN  LET key=18
590 IF qq=13 THEN  LET key=19
600 RETURN
610 REM
620 REM KEYBOARD
630 REM
640 PAPER 2: CLS : INK 0
650 LET a$="
660 PAPER 7
670 FOR i=10 TO 18
680 PRINT  AT i,6;a$
690 NEXT i
700 FOR i=64 TO 204 STEP 16
710 PLOT i,24: DRAW 0,71
720 NEXT i
730 LET a$="  "
740 RESTORE 750
750 DATA 9,11,13,17,19,23
760 FOR i=1 TO 6
770 READ a
780 FOR j=10 TO 14
790 PRINT  AT j,a;a$
```

```
800 NEXT j: NEXT i
810 LET a$="ERTUIP": INK 7: PAPER 2
820 RESTORE 750
830 FOR i=1 TO 6: READ a
840 PRINT  AT 9,a;a$(i): NEXT i
850 LET a$="ASDFGHJKLE"
860 FOR i=1 TO 10
870 PRINT  AT 19,i*2+5;a$(i)
880 NEXT i
890 RETURN
900 REM POKE_W 23672,0
910 POKE 23672,0
920 POKE 23673,0
930 RETURN
940 FOR k=1 TO 50
950 NEXT k
960 RETURN
990 REM
1000 REM PRACTISE
1010 REM
1020 GO SUB 620
1030 PRINT  AT 2,2;"PRACTISE: Press '0' to stop"
1040 GO SUB 400
1050 IF key=255 THEN  RETURN
1060 IF key<254 THEN  BEEP .025,key
1070 GO TO 1040
1080 REM
2000 REM RECORD
2010 REM
2020 GO SUB 620
2030 PRINT  AT 2,0;"RECORD: Press '0' to start and
stop"
2040 LET add=start
2050 GO SUB 400
2060 IF key <> 255 THEN  GO TO 2050
2070 GO SUB 940: GO SUB 900
2080 LET z=254
2090 GO SUB 400
2100 IF key<254 THEN  BEEP .025,key
2110 IF z=key THEN  GO TO 2090
2120 POKE add, PEEK 23672
2130 POKE add+1, PEEK 23673
2140 POKE add+2,z
2150 IF z=255 THEN  LET tune=1: RETURN
2160 LET z=key: LET add=add+3
2170 GO SUB 900: GO TO 2090
2200 REM
3000 REM PLAYBACK
3010 REM
3020 IF tune=0 THEN  RETURN
3030 GO SUB 620
3040 LET begin=start
3050 LET end=add
3060 REM
3070 PRINT  AT 2,10;"PLAYBACK"
3090 LET add1=begin
3100 LET dur= PEEK add1+256* PEEK (add1+1)
3110 LET note= PEEK (add1+2)
3120 IF note=255 OR add1 >= end THEN  PRINT  AT 2,10;"
": RETURN
3130 IF note=254 THEN  PAUSE dur
3140 IF note<254 THEN  BEEP dur/50,note
3160 LET add1=add1+3
3170 GO TO 3100
3180 REM
4000 REM SAVE
4010 REM
4020 IF tune=0 THEN  RETURN
4030 CLS
4040 INPUT "SAVE: Enter file name ";a$
4050 SAVE a$ CODE start,add-start+5
4060 RETURN
4070 REM
5000 REM LOAD
5010 REM
5020 CLS
5030 INPUT "LOAD: Enter file name ";a$
5040 LOAD a$ CODE start
5050 LET tune=1
5060 LET add=start
5070 IF  PEEK (add+2)=255 THEN  RETURN
5080 LET add=add+3
5090 GO TO 5070
5100 REM
5200 REM GET VALUES
5210 INPUT "Enter new note value ";note
5220 INPUT "Enter new duration ";dur
5230 IF note=254 THEN  GO TO 5250
5240 IF note>20 OR note <>  INT note THEN  GO TO 5210
5250 IF dur<1 OR dur <>  INT dur THEN  GO TO 5210
5260 RETURN
5270 REM
6000 REM EDIT
6010 REM
6020 IF tune=0 THEN  RETURN
6030 LET pages= INT ((add-start-2)/30)+1: LET item=1
6040 CLS : PRINT  AT 2,2;"EDIT": LET number=0
6050 PRINT  AT 4,2;"SELECT PAGE (1";
6060 IF pages=1 THEN  PRINT ")"
6070 IF pages>1 THEN  PRINT " to ";pages;")"
6080 INPUT "Enter page number ";page
6090 IF page<1 OR page>pages OR page <>  INT page THEN  GO TO
6080
6100 LET begin=start+30*(page-1)
6200 CLS : PRINT  AT 2,2;"EDIT": PRINT : PRINT
6210 FOR i=0 TO 9
6220 IF begin+i*3 >= add THEN  GO TO 6270
6230 LET number=number+1
6240 PRINT  TAB 4;i+1; TAB 8; PEEK (i*3+2+begin);
6250 PRINT  TAB 16; PEEK (i*3+begin)+256* PEEK (i*3+1+begin)
6260 NEXT i
6270 INK 0: PRINT  AT item+4,1;">"
6280 PRINT  AT 16,4;"P A I D E ^ v": INK 7
6290 LET z$= INKEY$
6300 IF z$="" THEN  GO TO 6290
6310 IF z$="p" THEN  GO TO 6400
6320 IF z$="a" THEN  GO TO 6450
6330 IF z$="i" THEN  GO TO 6520
6340 IF z$="d" THEN  GO TO 6640
6350 IF z$="7" OR  CODE z$=11 THEN  GO TO 6730
6360 IF z$="6" OR  CODE z$=10 THEN  GO TO 6800
6370 IF z$="e" THEN  RETURN
6380 GO TO 6290
6390 REM
6400 LET end=begin+3*number
6410 GO SUB 3060
6420 GO TO 6200
6440 REM
6450 GO SUB 5200
6460 LET edit=begin+3*(item-1)
6470 POKE edit,dur-256* INT (dur/256)
6480 POKE edit+1, INT (dur/256)
6490 POKE edit+2,note
6500 GO TO 6200
6510 REM
6520 PRINT  AT 21,4;"INSERT"
6530 GO SUB 5200
6540 LET add=add+3
6550 FOR i=add TO begin+3*(item) STEP -1
6560 POKE i, PEEK (i-3)
6570 NEXT i
6580 LET edit=begin+3*(item-1)
6590 POKE edit+1, INT (dur/256)
6600 POKE edit,dur-256* INT (dur/256)
6610 POKE edit+2,note
6620 GO SUB 6200
6630 REM
6640 PRINT  AT 21,4;"DELETE - Are you sure?"
6650 GO SUB 940: LET w$= INKEY$ : IF w$="" THEN  GO TO 6650
6660 IF w$ <> "y" THEN  GO TO 6200
6670 LET add=add-3
6680 FOR i=begin+3*(item-1) TO add
6690 POKE i, PEEK (i+3)
6700 NEXT i
6710 GO TO 6200
6720 REM
6730 PRINT  AT item+4,1;" "
6740 INK 0: LET item=item-1
6750 IF item=0 THEN  LET item=number
6760 PRINT  AT item+4,1;">"
6770 INK 7: GO TO 6290
6780 REM
6800 PRINT  AT item+4,1;" "
6810 INK 0: LET item=item+1
6820 IF item>number THEN  LET item=1
6830 PRINT  AT item+4,1;">"
6840 INK 7: GO TO 6290
6990 REM
7000 REM EXIT
7010 REM
7020 IF tune=0 THEN  STOP
7030 CLS : INPUT "EXIT: Send data to printer? ";a$
7040 IF a$="" THEN  LET a$="n"
7050 IF a$(1)="y" THEN  LET print=1
7060 IF a$(1) <> "y" THEN  LET print=0
7080 FOR i=start TO add+2
7090 LET x= PEEK i
7100 PRINT x;",";
7110 IF print THEN  LPRINT x
7120 NEXT i
7130 STOP
```
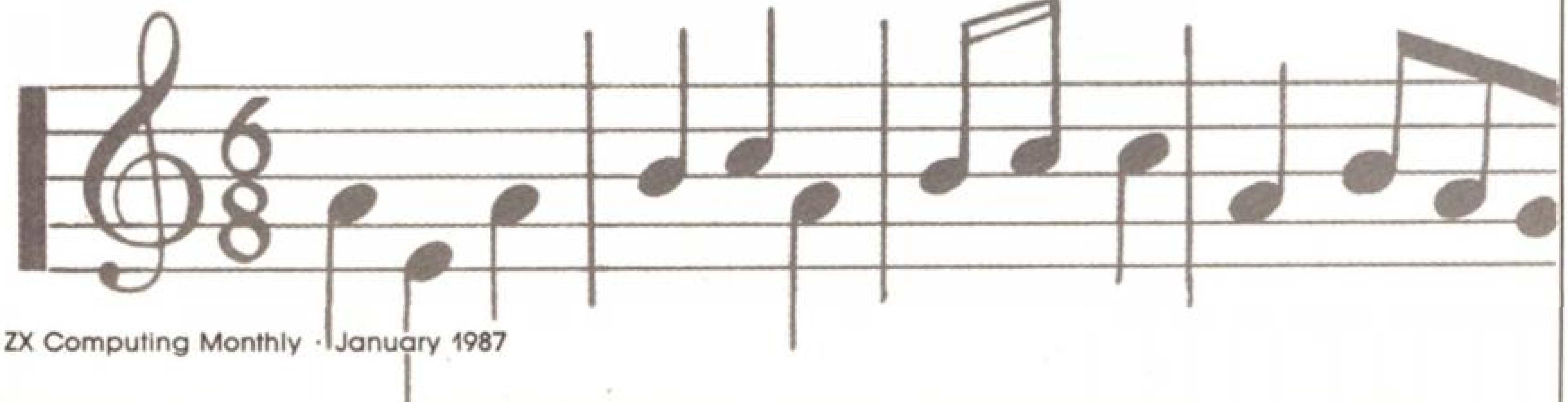


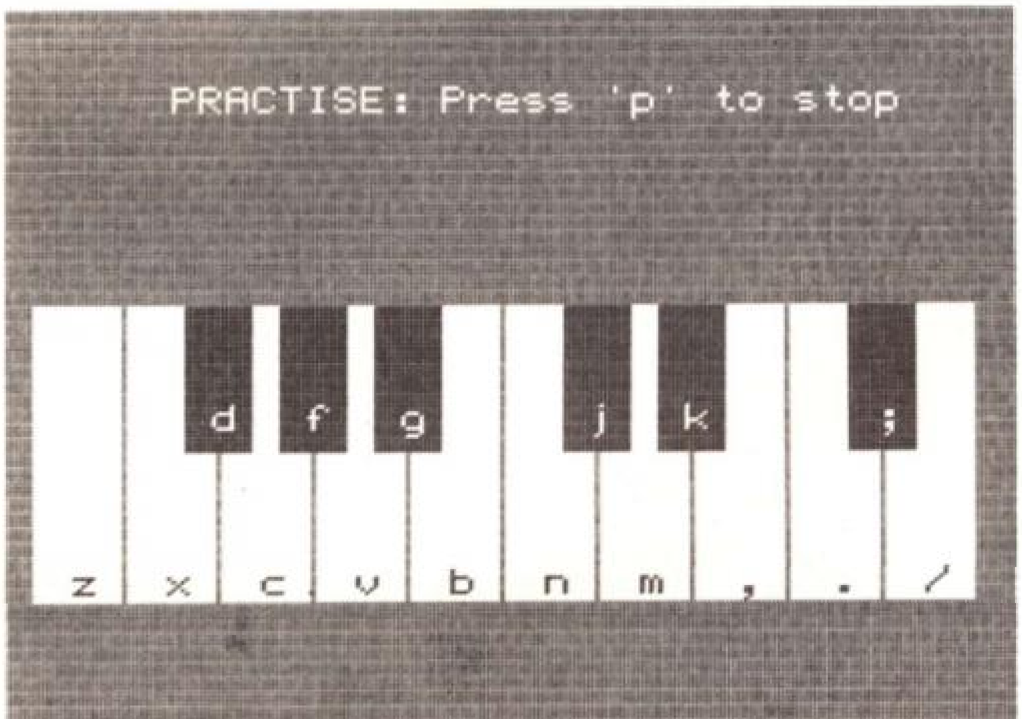
Figure 2

Tune Maker — QL Listing

```
100 REMark  *** Composer ***
110 REMark   David Nowotnik
120 REMark   October, 1986
130 :
140 INIT
150 REPeat main_loop
160   z=MENU
170   SELect ON z
180   =1: PRACTISE
190   =2: RECORD: tune=1
200   =3: PLAYBACK start+4,add
210   =4: SAVE_TUNE
220   =5: LOAD_TUNE
230   =6: EDIT_TUNE1
240   =7: EXIT_TUNE
250   END SELect
260 END REPeat main_loop
270 :
280 DEFine PROCedure TV
290 MODE 8: WINDOW 512,256,0,0
300 PAPER 0: CLS
310 WINDOW 448,200,32,16
320 WINDOW #2,448,200,32,16
330 WINDOW #0,448,40,32,216
340 PAPER 2: PAPER #2,1: PAPER #0,0
350 INK 7: INK #2,7: INK #0,7
360 CLS : CLS #0
370 END DEFine TV
380 :
390 DEFine PROCedure INIT
400 TV
410 start= RESPR(0)
420 IF start>221184 THEN start=RESPR(start-221184)
430 tune=0
440 END DEFine INIT
450 :
460 DEFine FuNction MENU
470 LOCal z,z$
480 CLS: INK 7: CSIZE 2,0
490 AT 2,16: PRINT "TUNE MAKER"
500 AT 4,8: PRINT "SELECT"
510 PRINT \\,"   1. PRACTISE"
520 PRINT,"   2. RECORD"
530 PRINT,"   3. PLAYBACK"
540 PRINT,"   4. SAVE"
550 PRINT,"   5. LOAD"
560 PRINT,"   6. EDIT"
570 PRINT,"   7. EXIT"
580 REPeat menu_loop
590   z$=INKEY$(-1)
600   IF z$>"0" AND z$<"8" THEN EXIT menu_loop
610 END REPeat menu_loop
620 z=z$: CLS: RETurn z
630 END DEFine MENU
640 :
650 DEFine FuNction WHICH_KEY
660 LOCal z$,i,key,x
670 FOR i=7,4,3,2
680  x=KEYROW(i): IF x THEN EXIT i
690 END FOR i
700 SELect ON i
710   =7: SELect ON x
720       =8: key=22
730       =16: key=15
740       =32: key=4
750       =64: key=11
760       =128: key=7
770       = REMAINDER : key=0
780       END SELect
790   =4: SELect ON x
800       =32: key=30
810       =64: key=20
820       =128: key=10
830       = REMAINDER : key=0
840       END SELect
850   =3: SELect ON x
860       =4: key=8
870       =16: key=16
880       =64: key=13
890       =128: key=5
900       = REMAINDER : key=0
910       END SELect
920   =2: SELect ON x
930       =2: key=24
940       =4: key=6
950       =8: key=18
960       =16: key=12
970       =64: key=9
980       = REMAINDER : key=0
990       END SELect
1000    = REMAINDER : key=0
1010 END SELect
1020 RETurn key
1030 END DEFine WHICH_KEY
1040 :
1050 DEFine PROCedure PRACTISE
1060 LOCal z
1070 KEYBOARD
1080 AT 2,6: PRINT "PRACTISE: Press 'p' to stop"
1090 REPeat practise_loop
1100   z= WHICH_KEY
1110   IF z=30 THEN EXIT practise_loop
1120   IF z>0 THEN
1130     BEEP 6000,z
1140   ELSE
1150     BEEP
1160   END IF
1170 END REPeat practise_loop
1180 END DEFine PRACTISE
1190 :
1200 DEFine PROCedure KEYBOARD
1210 LOCal i,x,a$
1220 x=12: a$="zxcvbnm,./"
1230 INK 0: PAPER 7
1240 FOR i=1 TO 10
1250   BLOCK 40,80,x,80,7
1260   CURSOR x+16,150
1270   PRINT a$(i)
1280   x=x+42
1290 END FOR i
1300 a$="sdfghjkl;"
1310 INK 7: PAPER 0
1320 FOR i=1,2,3,5,6,8
1330   x=i*42+38
1340   BLOCK 30,40,x,80,0
1350   CURSOR x+10,106
1360   PRINT a$(i+1)
1370 END FOR i
1380 INK 7: PAPER 2
1390 END DEFine
1400 :
1410 DEFine PROCedure RECORD
1420 LOCal z,key
1430 KEYBOARD
1440 AT 2,2: PRINT "RECORD: Press 'p' to start and stop"
1450 REPeat start_loop
1460    z=WHICH_KEY
1470    IF z=30 THEN EXIT start_loop
1480 END REPeat start_loop
1490 add=start: POKE_W 163886,0
1500 POKE_W add,0: POKE add+2,0
1510 key=0: PAUSE 10: add=add+4
1520 REPeat record_loop
1530   z=WHICH_KEY
1540   IF z=30 OR PEEK_W (163886)>65000 THEN EXIT record_loop
1550   IF z>0 THEN
1560     BEEP 10000,z
1570   ELSE : BEEP
1580   END IF
1590   IF z<>key THEN
1600     POKE add+2,z: POKE_W add, PEEK_W (163886)
1610     add=add+4: key=z: POKE_W 163886,0
1620   END IF
1630 END REPeat record_loop
1640 POKE_W add,PEEK_W (163886): POKE add+2,255
1650 END DEFine RECORD
1660 :
1670 DEFine PROCedure PLAYBACK (begin, ending)
1680 LOCal add1,z
1690 IF NOT tune THEN RETurn
1700 AT 2,12: PRINT "PLAYBACK"
1710 add1=begin
1720 REPeat playback_loop
1730   POKE_W 163886,0
1740   REPeat delay
1750     IF PEEK_W (163886)>= PEEK_W (add1) THEN EXIT delay
1760   END REPeat delay
1770   z=PEEK (add1+2)
1780   SELect ON z
1790   =0: BEEP
1800   =255: EXIT playback_loop
1810   = REMAINDER : BEEP 10000,z
1820   END SELect
1830   add1=add1+4
1840   IF ending<=add1 THEN EXIT playback_loop
1850 END REPeat playback_loop
1860 END DEFine PLAYBACK
1870 :
1880 DEFine PROCedure SAVE_TUNE
1890 LOCal a$
1900 IF NOT tune THEN RETurn
1910 AT 2,2: INPUT "Enter file name ";a$
1920 DELETE "mdv2_"&a$
1930 SBYTES "mdv2_"&a$,start,add+5-start
1940 END DEFine SAVE_TUNE
1950 :
1960 DEFine PROCedure LOAD_TUNE
1970 LOCal a$
1980 AT 2,2: INPUT "Enter file name ";a$
1990 LBYTES "mdv2_"&a$,start
2000 tune=1
2010 add=start
2020 REPeat find_end
2030   IF PEEK (add+2)=255 THEN EXIT find_end
2040   add=add+4
2050 END REPeat find_end
2060 END DEFine LOAD_TUNE
2070 :
2080 DEFine PROCedure EDIT_TUNE (begin)
2090 LOCal number,edit_item,z,i
2100 REPeat edit_loop1
2110   CLS: INK 7: AT 2,2: PRINT "EDIT"\\: CLS #0
2120   number=0: edit_item=1
2130   FOR i=0 TO 9
2140     IF begin+i*4>=add THEN EXIT i
2150     number=number+1
2160     PRINT TO 4;i+1;TO 8;PEEK (i*4+2+begin);
2170     PRINT TO 16; PEEK_W (i*4+begin+4)
2180   END FOR i
2190   AT 3+edit_item,1: INK 0: PRINT ">"
```

```
2200    AT 16,12: PRINT "P A I D E ^ v"
2210    REPeat edit_loop2
2220      REPeat edit_loop3
2230        z= CODE (INKEY$(-1))
2240        FOR i=112,97,105,100,101,208,216
2250          IF i=z THEN EXIT edit_loop3
2260        END FOR i
2270      END REPeat edit_loop3
2280      SELect ON z
2290      =112: PLAYBACK begin,number*4+begin
2300      AT 2,12: PRINT "             "
2310      =208: AT edit_item+3,1: PRINT " "
2320      edit_item=edit_item-1
2330      IF edit_item=0 THEN edit_item=number
2340      AT edit_item+3,1: PRINT ">"
2350      =216: AT edit_item+3,1: PRINT " "
2360      edit_item=edit_item+1
2370      IF edit_item>number THEN edit_item=1
2380      AT edit_item+3,1: PRINT ">"
2390      =101: EXIT edit_loop1
2400      =97: GET_VALUES
2410      edit_it=begin+4*(edit_item)
2420      POKE edit_it-2,note
2430      POKE_W edit_it, dur
2440      EXIT edit_loop2
2450      =105: PRINT #0,"  INSERT": GET_VALUES
2460      add=add+4
2470      FOR i=add TO begin+4*(edit_item) STEP -1
2480        POKE i, PEEK (i-4)
2490      END FOR i
2500      POKE begin-2+edit_item*4,note
2510      POKE_W begin+4*edit_item,dur
2520      EXIT edit_loop2
2530      =100: PRINT #0,"  DELETE"
2540      add=add-4
2550      FOR i=begin-2+edit_item*4 TO add
2560        POKE i, PEEK (i+4)
2570      END FOR i
2580      EXIT edit_loop2
2590 END SELect
2600   END REPeat edit_loop2
2610 END REPeat edit_loop1
2620 END DEFine EDIT_TUNE
2630 :
2640 DEFine PROCedure GET_VALUES
2650 REPeat values_loop
2660   INPUT #0,"  Enter new note value ";note
2670   INPUT #0,"  Enter new duration   ";dur
2680   IF note>0 AND note<30 THEN EXIT values_loop
2690   IF dur>=0 AND dur=INT(dur) THEN EXIT values_loop
2700   PRINT #0,"  Value incorrect - re-enter"
2710 END REPeat values_loop
2720 END DEFine GET_VALUES
2730 :
2740 DEFine PROCedure EDIT_TUNE1
2750 LOCal page,pages
2760 IF NOT tune THEN RETurn
2770 pages=INT ((add-start)/40)+1
2780 CLS: CLS #0: INK 7: AT 2,2: PRINT "EDIT TUNE"\\
2790 PRINT "   SELECT PAGE (1";
2800 IF pages=1 THEN
2810   PRINT ")"
2820 ELSE
2830   PRINT " to ";pages;")"
2840 END IF
2850 REPeat get_page
2860   INPUT #0,"Enter page number ";page
2870   IF page>0 AND page<=pages AND page=INT (page) THEN
EXIT get_page
2880 END REPeat get_page
2890 EDIT_TUNE (start+40*(page-1))
2900 END DEFine EDIT_TUNE1
2910 :
2920 DEFine PROCedure EXIT_TUNE
2930 LOCal z$,name$,file$,line_no,count,item,a$,data_item
2940 CLS: INK 7: CLS #0
2950 AT 2,2: PRINT "EXIT"\\
2960 IF NOT tune THEN STOP
2970 PRINT "   SAVE TUNE AS DATA? (y/n)"
2980 REPeat get_answer
2990   z$=INKEY$(-1)
3000   IF z$="n" OR z$="y" THEN EXIT get_answer
3010 END REPeat get_answer
3020 IF z$="n" THEN STOP
3030 PRINT \\"   PLACE A CARTRIDGE IN mdv2"\\
3040 INPUT "   Enter a file name ";name$
3050 file$='mdv2_'&name$
3060 DELETE file$: OPEN_NEW #5,file$
3070 line_no=1000: item=0
3080 REPeat exit_loop1
3090   count=0: a$=line_no&' DATA
3100   REPeat exit_loop2
3110     data_item=start+4*item
3120     IF data_item>add THEN EXIT exit_loop1
3130     a$=a$& PEEK (data_item+2)&','& PEEK_W
(data_item+4)&','
3140     item=item+1: count=count+1
3150     IF count=4 THEN EXIT exit_loop2
3160   END REPeat exit_loop2
3170   a$=a$(1 TO (LEN (a$)-1))
3180   PRINT #5,a$
3190   IF data_item>add THEN EXIT exit_loop1
3200   count=0: line_no=line_no+10
3210 END REPeat exit_loop1
3220 CLOSE #5
3230 STOP
3240 END DEFine EXIT_TUNE
```

Figure 3: Screen display of QL version in 'practice' mode



RUNning the program, the first thing you should see is the main menu giving seven options. And from the menu, you should be able to appreciate all the facilities that the program provides; you can practise playing a tune, record it, play back that recording, save and load data, and edit that data.

Black and white

So, selecting option one gives you the screen display shown in **fig. 3** (on the QL; it's slightly different on the Spectrum). On the QL, the 'white' keys are played using the bottom row of alphabetic keys ('z' to '/'), and 'black' keys use the keys on the next row up ('d' to ';'). To achieve the same range, 'white' keys are
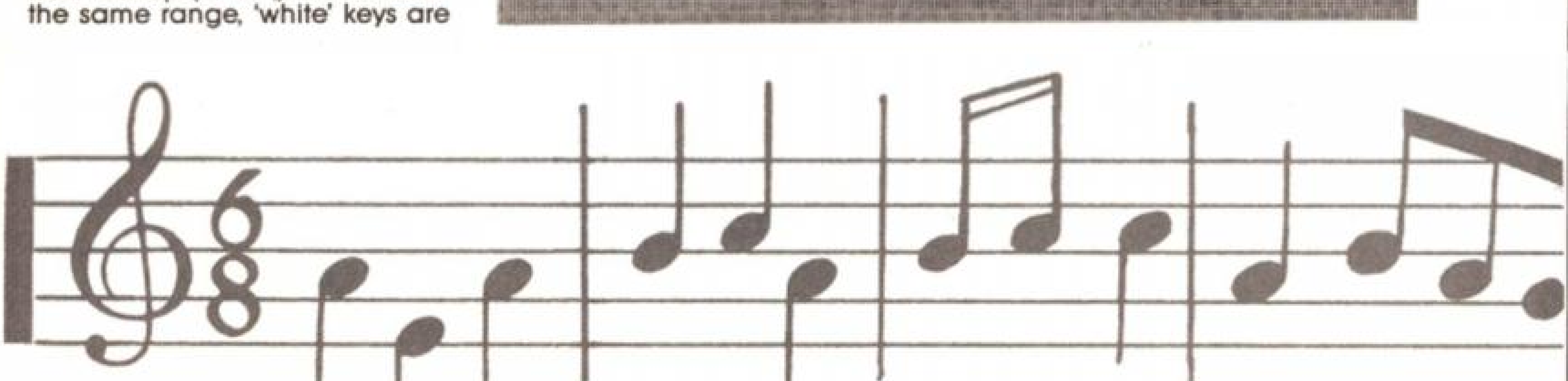


Figure 4: QL and Spectrum routines to play tunes in user programs

a) Spectrum Listing

```
2000 REM TUNE
2010 REM
2020 RESTORE 1000
2030 READ a,b,c
2040 IF c=255 THEN  RETURN
2050 LET dur=a+256*b
2060 IF c=254 THEN  PAUSE dur
2070 IF c<254 THEN  BEEP dur/50,c
2080 GO TO 2030
```

'a' to 'ENTER' and 'black' are 'e' to 'p' on the Spectrum. On both computers, the screen display reminds you of the effective keys, and their position on the musical keyboard.

In practise mode, press the keys to play a tune. Press only one at a time; unfortunately BEEP is only monophonic! On the QL, you'll hear a continuous note every time you press a key. But on the Spectrum, the note will be 'broken'. This difference is because of the way both computers generate sound. On the Spectrum, it is the main Z80 processor which generates the BEEP. It cannot be doing this and checking to see if a key is pressed, so the broken note you'll hear (in practise and record modes) will be the computer switching back and forth between generating sound, and checking for a keypress. On the QL, the main processor hands the sound generation over to a 'slave' processor, so sound generation and keypress checks can be carried out at the same. Playback of sound isn't limited in this way, so the music should sound smooth on both computers!

c) QL BEEP values

Note	BEEP value
E	24
F	22
F#	20
G	18
G#	16
A	15
A#	13
B	12
C	11
C#	10
D	9
D#	8
E	7
F	6
F#	5
G	4

So, once you practised a tune, and believe you are ready to record it, exit from practise mode (press '0' on the Spectrum, 'p' on the QL), and select option '2' on the main menu. Pressing 'O' on Spectrum, 'p' on QL starts and stops the recording session.

To listen to the result of your recording, press option '3' on the main menu.

If it's perfect, then you need not do much more (which I'll come to later), but you may want to make some changes. This is where option '6' comes in; the 'edit' option.

The recorded tune is held as a set of byte values above.a lowered RAMTOP. For each note of the active note, press 'a' (for alter), and you'll get a prompt to enter two new values.

If you wish to delete the 'active' note, press 'd'. To insert a new note (or pause) after the active note, press 'i'. You'll get the same prompt, for new duration and note values, as you got with the alter option. Pressing 'e' takes you back to the main menu.

Fine tuning

So, using the record and edit facilities, you should eventually get your tune just right. If the process is taking a little longer

b) QL Listing

```
800 DEFine PROCedure TUNE
810 RESTORE 32767
820 REPeat tune_loop
830    POKE_W 163886,0
840    READ a: IF a=255 THEN RETurn
850    READ b
860    IF a=0 THEN
870       BEEP
880    ELSE
890       BEEP 10000,a
900    END IF
910    REPeat delay
920       IF PEEK_W (163886)>=b THEN EXIT delay
930    END REPeat delay
940    BEEP
950 END REPeat tune_loop
960 END DEFine TUNE
```

(or pause there is a duration value, and a note value stored in sequence. The note value for pause is 0 on the QL version, and 254 on the Spectrum. Otherwise, note values are the same as supplied to the BEEP command (so note values are always different on both computers for the same note). In editing, you may want to change a note value; in this case refer to the manual (page 135) for changes to the Spectrum version, and to **fig. 4c,** where QL note values are listed.

In edit mode, your tune is broken down into 'pages' of ten notes (and pauses). First, you are asked to select which page you want to work on, then you are presented with the appropriate page, which lists the ten note values and their duration in sequence. To let you know where you are in the tune you can press 'p' to play the sequence of notes shown.

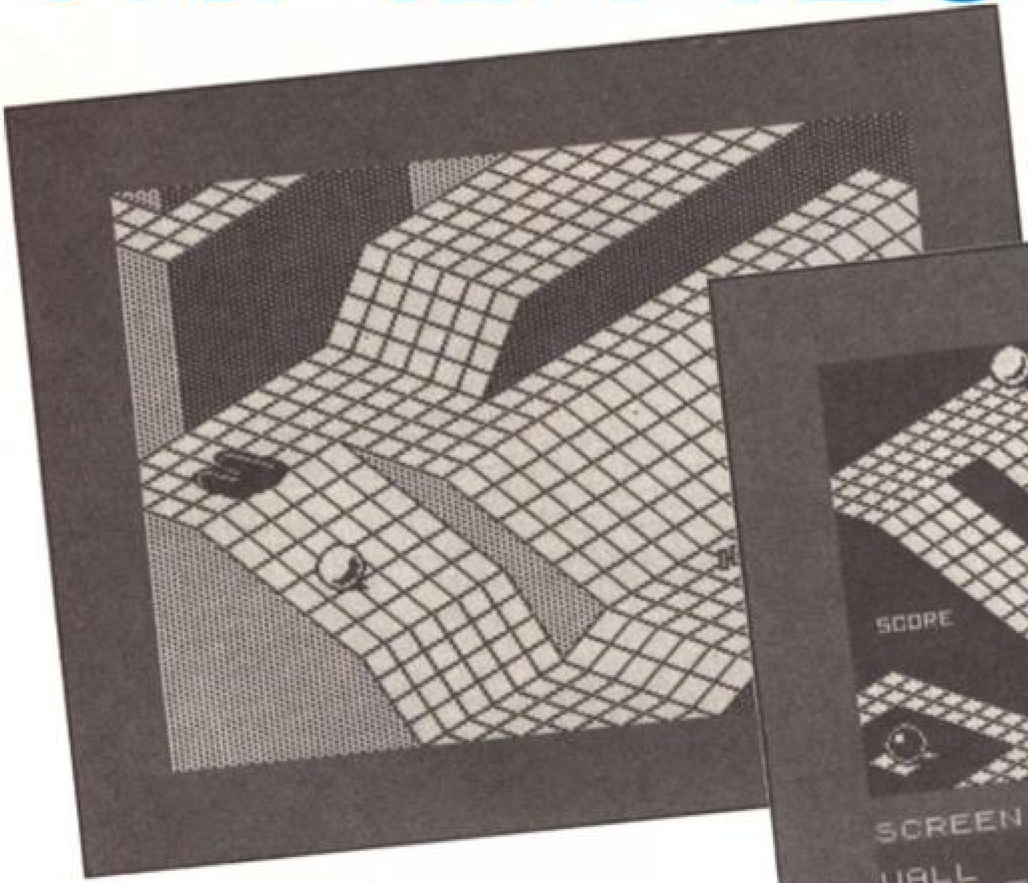
An arrow (>) appears beside the first note. This arrow shows you the 'active' note for editing. If you want to move to another note, press the up/down cursor keys (QL and Spectrum Plus) or '6' or '7' on the Spectrum. To alter duration and/or note value

than anticipated, you can always save the tune data to tape (Spectrum version) or mdv2_ (QL), then re-load it later, using options '4' and '5' on the main menu.

But once you have your masterpiece, then you'll wish to extract it from this program, to allow it to be inserted into one of your own programs. Option '7' — exit — on the main menu deals with this. On the QL version, the data will be automatically fed into DATA lines and saved to mdv2_ under a file name of your choice. In **fig. 4b,** you'll see a QL procedure called 'TUNE'. Use this in your program; MERGE the tune DATA lines, so that, whenever the procedure TUNE is called, your masterpiece will be played.

On the Spectrum version, the numbers which should go into the DATA lines are listed on the screen, or can be dumped to a ZX printer. You'll then have to type in your own DATA lines (start at line number 1000) with these numbers in sequence. Add them to your Spectrum tune subroutine, listed in **fig. 4a,** so when you GO SUB 2000, your tune will be played.

WIN MARBLE MADNESS!





Win Melbourne House's Marble Madness Construction Set but beware of losing your marbles in our brain teaser puzzle.

Marble Madness, an arcade classic, has now been converted to the Spectrum and although the game itself could drive you to distraction there are additional opportunities to send yourself crazy by devising your own Marble Madness arena with the construction set.

The construction set feature means you need never play the same game twice and you can put together obstacle courses as simple or as complex as you like.

Break the code

Twenty-five copies of Marble Madness construction set are up for grabs and to win you must solve the three code puzzles on the entry coupon and decide which number comes next in the series.

Send your entries to Marble Madness Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.

The competition is open to all ZX readers except employees of Argus Specialist Publications, Chase Web and Melbourne House.

The editor's decision is final and no correspondence can be entered into. Please remember to write your answers on the back of the envelope.

The closing date is February 10th 1987.

Marble Madness Competition

What is the next number in the series?

1) 0123402341034120? ______

2) 412443014442101? ______

3) 0401431240310? ______

Name ....

Address ....

....

....

Send your entry to Marble Madness Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.



Illuminator

Alan Davis unveils a versatile character designer and outlines the first step towards illuminated text displays.

Even if you have only a slight interest in the world of adventure games, you could hardly have failed to encounter the great "TEXT" versus "GRAPHICS" debate. To illustrate or not to illustrate — that is the question! At one extreme we have the "text-only" purist who dismisses the notion of "graphic-enhanced" games on principle, and at the other extreme we find the philosophy that "a picture is worth a thousand words" — and so the fewer words the better!

Most of us, I suspect, fall somewhere in the middle — because naturally the situation isn't as simple as the extremists would have us believe. When you play a well-written text adventure such as "Castle Blackstar", with its evocative descriptions and intricate plot, who needs pictures? Contrariwise, who would now want to play a version of "The Hobbit" without those wonderfully atmospheric illustrations? The loss would be intolerable.

In a world where merit is always recognised and rewarded, there would be room for all styles to rub along together, but alas — we don't live in such a world. Ever since the tablet of stone was found on some remote mountain top, inscribed with the commandment "GRAPHIC ADVENTURES SELL BEST" on one side and "ANY OLD GRAPHICS WILL DO" on the other, we've seen a rapid decline in the development of the pure text game. And one unfortunate consequence of this is that, like it or not, the adventurer's Spectrum now spends a fair bit of time (and lots of memory) displaying pretty dreadful pictures. (Did you ever meet anyone who wasn't grateful that the graphics in "Emerald Isle" can be switched off?)

Text effects

Now this is a great shame, because the potential of the pure text adventure is far from being played out. But even the most diehard text purist must surely admit that a TV screen full of text isn't the most attractive sight around. The standard Spectrum character set is extremely crude and stark. (Imagine reading a novel which was printed using the Spectrum font! It doesn't bear thinking about, does it?) Of course many adventure authors use a redesigned character set for just this reason — though in a large number of cases even this concession to eye-appeal has been neglected.

Far back in the dim mists of time — about a couple of years ago — an adventure appeared under the Games Workshop banner which seemed to open up a new venue for the pure text game. It was called "Tower of Despair", and it was written (to the astonishment of all who saw it for the first time) with the Quill. It was a pretty good adventure in its own right, but what captured everyone's imagination was its presentation — which tried to produce the effect of illuminated manuscript using a combination of decorated initial letters and a very beautiful script. I understand that it was quite successful commercially at the time, but as far as I'm aware, no one has since developed the idea to any extent. (If you've found any others, perhaps you could drop me a line and let me know?)

Well, a two-year wait is long



enough; if no one else is going to try it, we'd better do it ourselves! After all, so many adventures are designed around a fantasy world with a medieval or Anglo-Saxon flavour, that we could hardly find a more appropriate kind of presentation than by simulating illuminated manuscript. Who knows — perhaps we can breathe a bit of life back into the old text adventure yet? (Of course the use of a display utility, even though we're starting with adventures in mind.)

Before tackling the actual computing problems, I went in search of inspiration by looking at reproductions of the genuine article. I spent some time poring over the Lindisfarne Gospels, which are masterpieces of illuminated script about 1200 years old. (You can see the original manuscripts in the British Museum). From this experience I can confidently say that if you were hoping to copy THAT on your Spectrum, then forget it! It was clear that I'd have to lower my sights quite a bit! However, eventually I did arrive at a number of definite and realistic program design aims which can be summarised as follows:

1. Tower of Despair was really a bit of a cheat; it had to be, simply because of the Quill's limitations. The effect of illuminated initials was achieved just by printing graphic symbols in various colours around an ordinary capital letter. Now we can do better than this because, free of the Quill's limiations, we can design large and detailed initial letter shapes, store them in memory, and then print them with a specially written machine code routine.

2. The illuminated initials should be large enough to produce their decorative effect — but not so large that either a ridiculous amount of memory is needed to store them, or that too little space is left on screen for the main body of text. I settled for squares 16 pixels (2 character squares) wide — so that 26 capital letters can be stored in a total of 832 bytes (26 letters at 32 bytes each).

3. The machine code printing routine should be readily accessible from within a BASIC program so that once a BASIC string variable (z$) has been defined, the first letter is automatically printed as an illuminated capital (with an option to switch to "normal" capitals if required). The routine should then continue to print the rest of the string in characters of normal size, word-wrapping at the ends of lines.

4. The word-wrapping process should be flexible enough to allow decorated borders of any width to be permanently displayed without the text over-

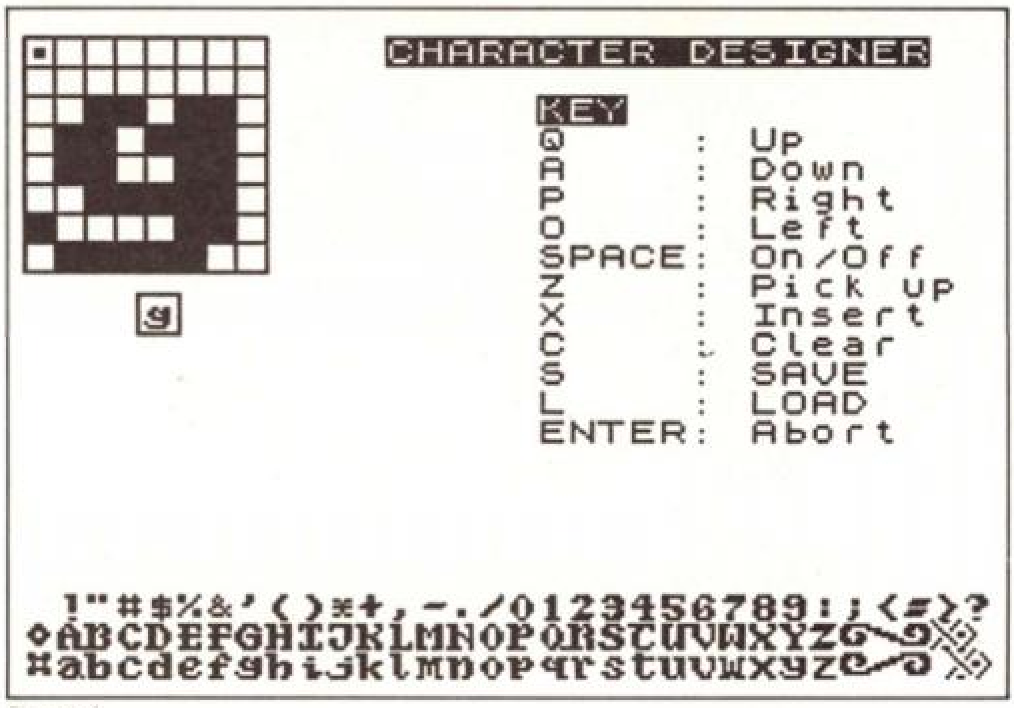


Figure 1

printing them; and for the same reason, a routine should be available for clearing selected areas of screen within any such borders.

5. It should be possible to alter the INK, PAPER, and BRIGHT attributes for the capitals separately from the main text.

This design brief would result in a flexible utility which could be used from within a BASIC adventure program with a minimum amount of fuss. Once the initial parameters are decided upon, such as width of decorated border and sizes of text windows, you can just get on with writing your game knowing that the bread-and-butter business of printing and arranging your text will be looked after automatically. If the end product falls a bit short of the Lindisfarne Gospels, it's nevertheless much more atmospheric and attractive than a typical screen from your average text adventure!

Capitals

Sounds promising? Then let's make a start. The first essential job is to design our 26 enlarged decorated initial letters within their 16x16 pixel squares, and store them as a series of bytes above RAMTOP. If you want to do this on graph paper, work out the numbers, and POKE them in yourself then fine. Off you go and we'll see you again in a year or two, perhaps . . . We mere mortals are going to use the ILLUMINATOR program in Listing 1, and do it all in a couple of hours instead! OK?

Listing 1, by the way, isn't just a utility for designing large decorated initials — it can also do service as an ordinary character designer at the press of a key. Of course you may well have a program for redesigning characters already, but since you'll need to change the normal character set to a style which "goes well" with your special capitals, it obviously makes sense to have the two facilities available together in a single program for both practical and aesthetic reasons.

To get a working copy of the program, type in Listing 1 and save it so that it will auto-run from line 1000. (I suggest you omit the REM statements in your version). Notice that lines 635, 640, and 1000 contain microdrive SAVE and LOAD commands (fast storage is a great advantage when working with this kind of utility). If you're restricted to tape cassettes, you'll need to change these to the usual tape commands, of course — which just means omitting the " *";1; " part in each case. Note also that in line 52 it is DOUBLE spaces which are used in the PRINT statements. The program isn't of much interest in itself — its value lies purely in its role as a development tool.

In addition to Listing 1, you'll also need a saved copy of the boring old Spectrum character set. To get one, just type in Listing 2 and then RUN. If you're working from tapes rather than drives, you'll need to change line 50, and then save the bytes immediately following the main program (Listing 1) so that they'll be loaded automatically (see line 1000).

Once you have the program loaded in and running, it will initially enter "CHARACTER DESIGNER" mode (Figure 1). If you press ENTER, it will go into "ILLUMINATOR" mode (Figure 2) — and indeed you can switch between the two modes at will at any time. Key identifications are given on screen (the same in both modes) but limited screen space requires these to be brief, and a little further explanation is called for. Movement of the little flashing cursor around the grid is

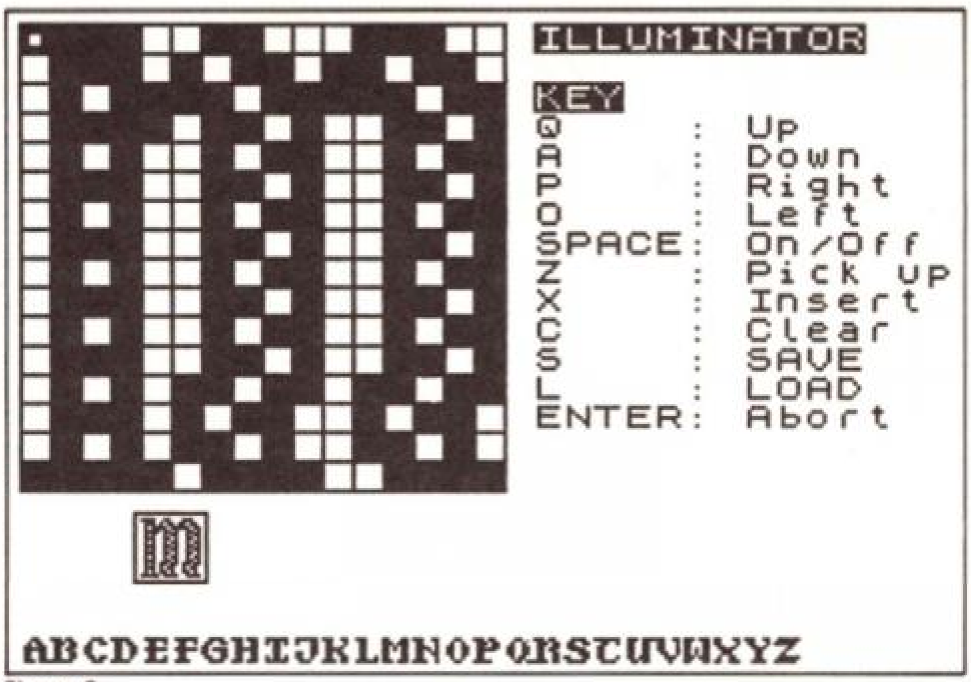


Figure 2

achieved using keys Q, A, O, and P. SPACE will toggle the pixel on and off at the current cursor position. The results of your artistic efforts will be shown at their normal size in the little box below the main grid, and if you get in such a mess that you need to start again, then key C will wipe the slate clean.

Illuminated cursor

When you're satisfied with your design, press key X to enter "insert" mode. A bright cursor will appear at screen bottom, and you can move it — again using keys Q, A, O, and P — to the letter or character you want the shape to represent. (The cursor will "wrap around" at the left and right edges to speed up the selection process). Pressing SPACE will insert the new shape, and ENTER allows you to quit if you change your mind. If you want to fiddle with an existing shape, you can transfer it to the grid by pressing key Z, and it's then a matter of moving the bright cursor to the letter or character you want to pick up, and pressing SPACE, in just the same way.

Once you've finished, you can save your shapes by pressing key S. You'll be asked to type a single identifying letter (say "a") so that your shapes can be saved as either "chars:a" CODE 64000,768 or "capitals:a" CODE 62976,832, depending on which mode you're in. Notice that you do need to save the normal characters and the big capitals separately — saving one does NOT save the other as well! You can load in a set of characters or capitals (to make additions or alterations) using key L — and again you'll be asked to specify an identifying letter. (If you already have a library of character sets saved under another name, and want to modify one of those from within ILLUMINATOR, then BREAK; load in your old character set bytes at 64000; and then restart with GOTO 1010).

In ILLUMINATOR mode there isn't room to display all the large initials on screen — and so the letters you see at screen bottom are just the normal capitals. However, you'll very reasonably want to review what you've done at intervals. To do this just press key Z and whizz the bright cursor along the row of letters. As the cursor passes over each letter in turn, the appropriate illuminated version will obligingly present itself in the small box for inspection.

Finally — perhaps for new readers, or those who've never felt inclined to try designing letter shapes before — if inspiration fails you, DON'T GIVE UP! I've given in Figure 3 some

Listing 1

```
   1 REM *** ILLUMINATOR ***
   2 REM
   3 REM *** PRINT LARGE CHARACTER ***
   4 REM
   5 IF mode AND chr>57 THEN  LET chr=32
   6 POKE 23607,245+(3 AND chr>55): LET tc=chr-(24 AND chr>55):
LET big=4*tc: PRINT AT 17,4;CHR$ (big-96);CHR$ (big-95);AT 18,4;
CHR$ (big-94);CHR$ (big-93): POKE 23607,60: RETURN
   7 REM
   8 REM *** HIGHLIGHT MENU SELECTION ***
   9 REM
  10 LET p=6+(1 AND p=6): PRINT AT pt+2,17; OVER 1; PAPER p,: RE
TURN
  17 REM
  18 REM *** SCAN KEYBOARD ***
  19 REM
  20 POKE 23658,0: PAUSE 20: LET i$=INKEY$: IF i$<>"" THEN  BEEP
 002,35: BEEP .002,40
  21 RETURN
  27 REM
  28 REM *** PRINT SCREEN TITLE AND MENU ***
  29 REM
  30 CLS : POKE 23607,60: PRINT AT 0,12+(5 AND mode); INVERSE 1;
;"CHARACTER DESIGNER" AND NOT mode;"ILLUMINATOR" AND mode;AT 2,1
7;"KEY"
  31 RESTORE 8010: PRINT AT 3,17;: FOR i=1 TO 11: READ x$: PRINT
 TAB 17;x$: NEXT i
  32 RETURN
  47 REM
  48 REM *** CLEAR GRID ***
  49 REM
  50 LET len=8+mode: DIM x(len,len): DIM p(8,1+(3 AND mode)): PR
INT AT 0,0;: FOR i=1 TO len: PRINT g$( TO len): NEXT i: PLOT 0,4
7+(64 AND NOT mode): DRAW 64+(64 AND mode),0:: DRAW 0,64+(64 AND
 mode)
  51 PLOT 30,41+(64 AND NOT mode): DRAW 11+mode,0: DRAW 0,-11-mo
de: DRAW -11-mode,0: DRAW 0,11+mode: IF NOT mode THEN  PRINT AT
9,4;" "
  52 IF mode THEN  PRINT AT 17,4;"  ";AT 18,4;"  "
  53 RETURN
  57 REM
  58 REM *** PRINT CHARACTER SET ***
  59 REM
  60 POKE 23607,249: IF NOT mode THEN  PRINT AT 19,0;k$
  61 IF mode THEN  PRINT AT 21,0;k$(34 TO 59)
  62 POKE 23607,60: RETURN
  97 REM
  98 REM *** ENLARGE SYMBOL ***
  99 REM
 100 DIM p(8,1+(3 AND mode)): DIM x(8+mode,8+mode): IF mode THEN
 GO SUB 5
 105 IF NOT mode THEN  POKE 23607,249: PRINT AT 9,4;CHR$ chr
 110 POKE 23607,60: OVER 1: FOR w=1 TO 1+(3 AND mode): LET ei=8
AND w>2: LET ej=8 AND (w=2 OR w=4): LET st=62975+(1024 AND NOT m
ode)+8*(w-1)+(8+(24 AND mode))*(chr-32): FOR i=1 TO 8: LET val=P
EEK (st+i): LET p(i,w)=val: LET n=256
 115 FOR j=1 TO 8: LET n=n/2: IF INT (val/n) THEN  LET x(i+ei,j+
ej)=1: PRINT AT i-1+ei,j-1+ej;CHR$ 145: LET val=val-n*INT (val/n
)
 120 NEXT j: NEXT i: NEXT w: OVER 0: RETURN
 127 REM
 128 REM *** INSERT SHAPE INTO MEMORY ***
```

```
 129 REM
 130 LET st=62975+(1024 AND NOT mode)+(8+(24 AND mode))*(chr-32)
 135 FOR w=1 TO (1+(3 AND mode)): FOR i=1 TO 8: POKE st+i+8*(w-1
),p(i,w): NEXT i: NEXT w: IF mode THEN  GO SUB 5
 140 IF NOT mode THEN  POKE 23607,249: PRINT AT oy,ox;CHR$ chr:
POKE 23607,60
 145 RETURN
 197 REM
 198 REM *** MAIN LOOP ***
 199 REM
 200 LET x=0: LET y=0
 210 PRINT AT y,x; OVER 1;CHR$ 144
 220 GO SUB 20
 225 PRINT AT y,x; OVER 1;CHR$ 144
 230 LET x=x+(i$="p" AND x<7+mode)-(i$="o" AND x>0)
 240 LET y=y+(i$="a" AND y<7+mode)-(i$="q" AND y>0)
 245 LET px=1+INT (x/8)+2*INT (y/8): LET py=1+y-(8 AND y>7)
 250 IF i$=CHR$ 32 THEN  PRINT  OVER 1;AT y,x;CHR$ 145: PLOT  OV
ER 1;32+x,39+(64 AND NOT mode)-y: LET pix=x(y+1,x+1): LET x(y+1,
x+1)=NOT pix: LET p(py,px)=p(py,px)+((-1 AND pix)+(1 AND NOT pix
))*2^(7-x+(8 AND x>7))
 260 IF i$="z" THEN  LET pt=6: GO SUB 10: GO SUB 400: GO SUB 10:
 IF o$<>CHR$ 13 THEN  GO SUB 50: GO SUB 100: GO TO 200
 270 IF i$="x" THEN  LET pt=7: GO SUB 10: GO SUB 400: GO SUB 10:
 IF o$<>CHR$ 13 THEN  GO SUB 130: GO TO 200
 280 IF i$="c" THEN  LET pt=8: GO SUB 10: GO SUB 50: GO SUB 10:
GO TO 200
 290 IF i$=CHR$ 13 THEN  LET mode=8 AND NOT mode: GO TO 1015
 300 IF i$="l" OR i$="s" THEN  LET pt=9+(i$="l"): GO SUB 10: GO
SUB 600: GO SUB 10: IF i$="l" THEN  GO TO 1010
 310 GO TO 210
 397 REM
 398 REM *** SELECT SYMBOL ***
 399 REM
 400 POKE 23607,60: LET xmax=(25 AND mode)+(31 AND NOT mode): IF
 mode THEN  LET oy=21: LET ox=ox AND ox<26: PRINT AT 17,9;"MOVE
CURSOR TO SYMBOL";AT 18,8; PAPER 6;"SPACE"; PAPER 7;"/SELECT; ";
 PAPER 6;"ENTER"; PAPER 7;"/QUIT": GO SUB 5
 405 IF NOT mode THEN  PRINT AT 16,5;"MOVE CURSOR TO SYMBOL";AT
17,1; PAPER 6;"SPACE"; PAPER 7;" TO SELECT, "; PAPER 6;"ENTER";
PAPER 7;" TO QUIT"
 410 PRINT AT oy,ox; OVER 1; BRIGHT 1;" "
 420 POKE 23658,0: PAUSE 0: LET o$=INKEY$: BEEP .005,40
 430 PRINT AT oy,ox; OVER 1; BRIGHT 0;" "
 440 IF NOT mode THEN  LET oy=oy-(o$="q" AND oy>19)+(o$="a" AND
oy<21)
 450 LET ox=ox+(xmax AND o$="o" AND ox=0)-(o$="o" AND ox>0)+(o$=
"p" AND ox<xmax)-(xmax AND o$="p" AND ox=xmax)
 460 LET chr=32+ox+(32*(oy-19) AND NOT mode): IF mode THEN  GO S
UB 5
 470 IF o$=CHR$ 32 OR o$=CHR$ 13 THEN  GO TO 500
 480 GO TO 410
 500 IF mode AND o$=CHR$ 13 THEN  GO SUB 52
 505 IF mode THEN  PRINT AT 17,7,,;AT 18,7,,: RETURN
 510 PRINT AT 16,0,,,,: RETURN
 597 REM
 598 REM *** LOAD AND SAVE ***
 599 REM
 600 POKE 23607,60: LET z$=("chars:" AND NOT mode)+("capitals:"
AND mode): LET start=62976+(1024 AND NOT mode): LET length=768+(
64 AND mode)
 610 PRINT #1;AT 1,0; PAPER 6;"IDENTIFYING LETTER FOR FILE?"
 620 POKE 23658,0: PAUSE 0: LET o$=INKEY$: IF o$=CHR$ 13 THEN  P
RINT #1;AT 0,0,,,,: RETURN
 625 IF o$<"a" OR o$>"z" THEN  GO TO 620
 630 LET z$=z$+o$: PRINT #1;AT 0,0,,,,AT 1,0;
 635 IF i$="s" THEN  SAVE *"m";1;z$CODE start,length
 640 IF i$="l" THEN  LOAD *"m";1;z$CODE start,length
 645 RETURN
 997 REM
 998 REM *** INITIALISATION ***
 999 REM
1000 CLEAR 62975: LOAD *"m";1;"specchars"CODE 64000: PRINT AT 10
,10;"PLEASE WAIT": LET chr=32: LET mode=0: RESTORE : FOR i=65368
 TO 65391: READ x: POKE i,x: NEXT i
1005 LET k$="": FOR i=32 TO 127: LET k$=k$+CHR$ i: NEXT i: LET g
$="": FOR i=1 TO 16: LET g$=g$+CHR$ 146: NEXT i
1010 LET oy=19: LET ox=0: LET p=7
1015 GO SUB 30: GO SUB 50: GO SUB 60: GO TO 200
7997 REM
7998 REM *** DATA STATEMENTS ***
7999 REM
8000 DATA 0,0,0,28,28,28,0,0,0,127,127,127,127,127,127,127,255,1
28,128,128,128,128,128,128,128
8010 DATA "Q     : Up"
8011 DATA "A     : Down"
8012 DATA "P     : Right"
8013 DATA "O     : Left"
8014 DATA "SPACE: On/Off"
8015 DATA "Z     : Pick up"
8016 DATA "X     : Insert"
8017 DATA "C     : Clear"
8018 DATA "S     : SAVE"
8019 DATA "L     : LOAD"
8020 DATA "ENTER: Abort"
```

suggested shapes for the enlarged capitals, printed off at double size — and if you're stuck for a set of vaguely "archaic" normal characters, you might get some ideas from Figure 1. If all else fails, you can always start with these. Incidentally, the funny shapes you see scattered around among the letters at the bottom of Figure 1 are designs I've experimented with in order to make decorative borders and text dividers. I'll be offering a few suggestions about that next month, but of course our main concern will be with the machine code routine which is going to transform this month's artistic dabblings into masterpieces of illuminated script . . .

Listing 2

```
 1 REM *** COPY SPECTRUM CHARACTER SET ***
10 CLEAR 63999
20 FOR i=1 TO 767
30 POKE 64000+i,PEEK (15616+i)
40 NEXT i
50 SAVE *"m";1;"specchars"CODE 64000,768
```



Figure 3

Part II next month

THE PROFESSIONAL TOUCH



Steve Turner with a step by step guide to the creation of his soon to be released game Ranarama.

Over the last few months I have covered various aspects of writing a game to professional standards. Well here's how I put the theory into practice developing my latest game Ranarama.

The specification

The main idea for the game came from Paradroid. I had already used the game play from Paradroid in Quazatron but with a very different presentation.

I had always thought that a fantasy magic came would work very well with the game plot and had quite a firm idea of what I wanted to program.

Here is a list of features I wanted in the game:

1. Bas relief light and shade graphics as used in Paradroid, much in vogue on the C64 but never really exploited on the Spectrum.
2. No scrolling so I could put lots of colour on the screen.
3. A plan view like Paradroid but on a different scale so several rooms would appear on one screen.
4. Game play similar to Quazatron but based on magic with a magic combat game as a sub game.
5. Hundreds of lesser meanies to zap.
6. My unique feature of the game, the maps and screens to be kept blank until they have been explored. Thus the screens gradually reveal themselves. I had this idea from testing Quazatron. If I cleared the screen test the plot buffers. I found it much fun to explore the screen in the dark with it gradually being coloured in as I moved around.
7. Hidden view processing so the player can only see what is happening in the current room. This idea came from Paradroid but was refined to save processing time. The advantage this feature offers is a smaller moving screen area.
8. Total use of the Spectrum screen. So many games seem to shrink the screen. I wanted to use 100 per cent of the screen for the rooms display so a feeling of location and scale would be given to the player.

   This meant no on-screen displays or messages. The sound and actions must be able to convey all important events while the game is running.
9. The place to be infested with

intelligent meanies who are all properly placed in the dungeon. They must be able to move from room to room off screen and try to find the player. They must enter and exit the current room properly, not just appear out of nowhere.

I felt the above formula would enable me to use the Spectrum's colour and graphics potential to the full while leaving CPU time to run a fast game.

Feasibility

Now it was time to run up a mock screen to test the basic look of the game. I used a character generator to build a screen and was really pleased with the solid effect of the walls. I then had to size the program to the Spectrum's RAM.

I used Quazatron as a guide, reserving areas for the size maps and graphics I required. I soon found, like always, my original ideas were a bit ambitious. I wanted 24 types of animated meany but had to chop this down to 14.

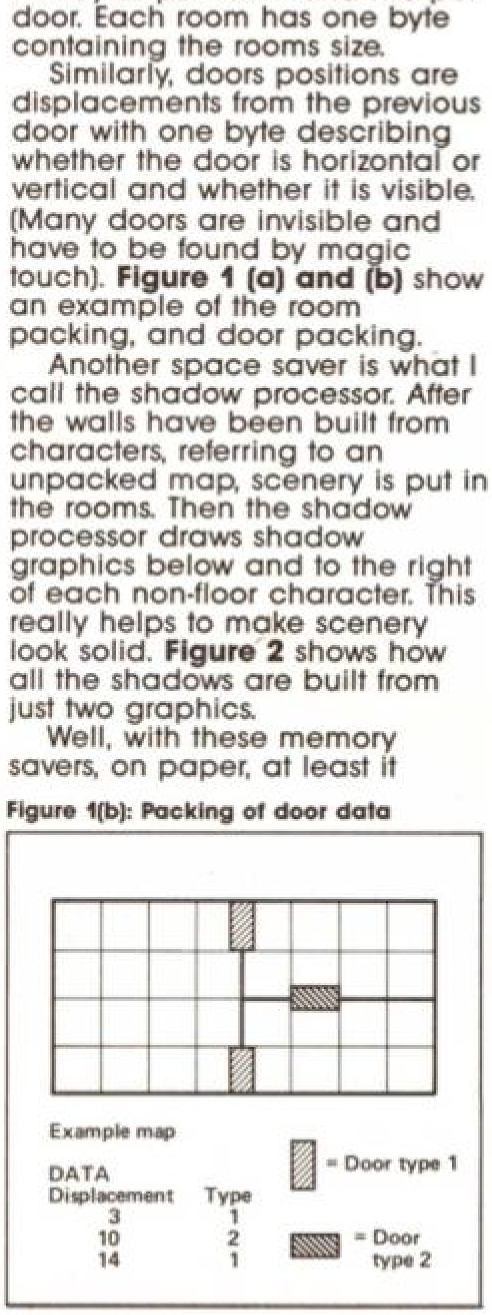
The map packing was essential as memory was looking short. I played around with a few methods and finally settled on a way of describing a map with two bytes per room and two per door. Each room has one byte containing the rooms size.

Similarly, doors positions are displacements from the previous door with one byte describing whether the door is horizontal or vertical and whether it is visible. (Many doors are invisible and have to be found by magic touch). **Figure 1 (a) and (b)** show an example of the room packing, and door packing.

Another space saver is what I call the shadow processor. After the walls have been built from characters, referring to an unpacked map, scenery is put in the rooms. Then the shadow processor draws shadow graphics below and to the right of each non-floor character. This really helps to make scenery look solid. **Figure 2** shows how all the shadows are built from just two graphics.

Well, with these memory savers, on paper, at least it

Figure 1(b): Packing of door data



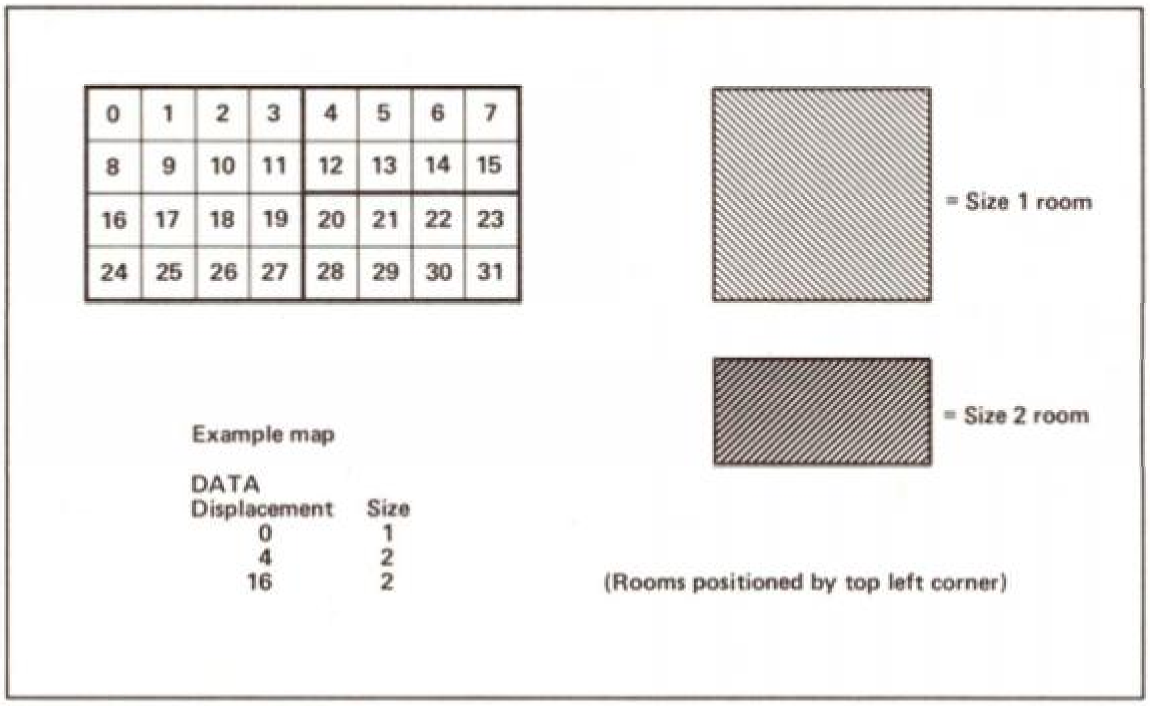


Figure 1(a): Packing of room data

could fit into the machine. I was worried about the complexity of the screen build and how long it would take. So I decided before going much further to code that routine. It proved a difficult job and took a month. Now it was time to structure the game before I was tempted to keep adding bits.

Structured design

In practice this turned out to be a formality. The top level structure was almost identical to Quazatron. **Figure 3** shows the actual structure I used.

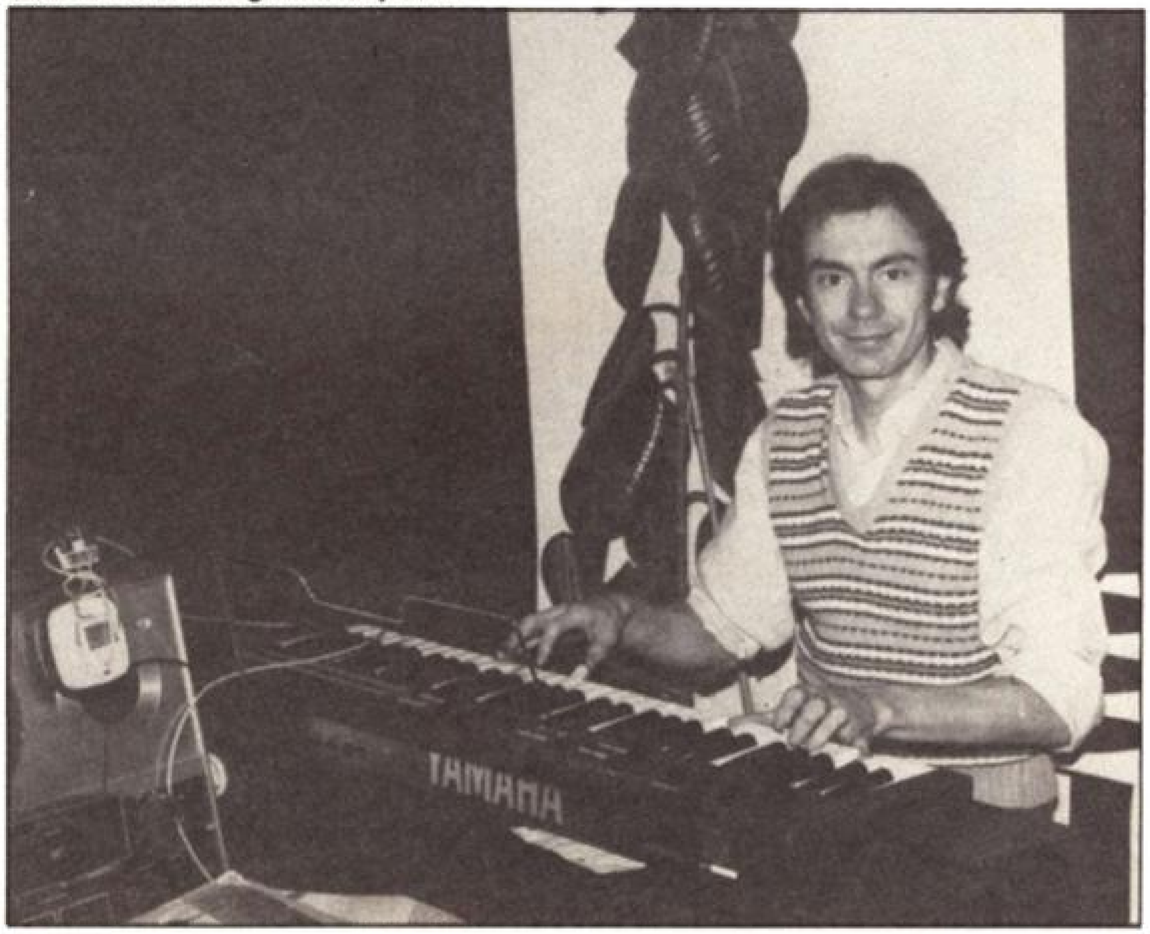
Coding the top level structure was a matter of taking Quazatron and chopping out unwanted bits. Now I had a framework to build upon.

Coding

The first task was to collect all the subroutines from one game to the next. There is no point in re-inventing code that you know works. It takes me eight man months to complete a game nowadays, but I do absolutely everything except the text character set. So it is extremely important to save time where possible or the price of the game would have to rise to cover the extra cost.

During coding I started to play with the graphics. It was then a crisis developed. I could not draw a decent wizard as the main graphic in the space I had allowed. So much for coding the complicated bit first. I was loathe to change it. Then

Steve Turner: relaxing at the keyboard



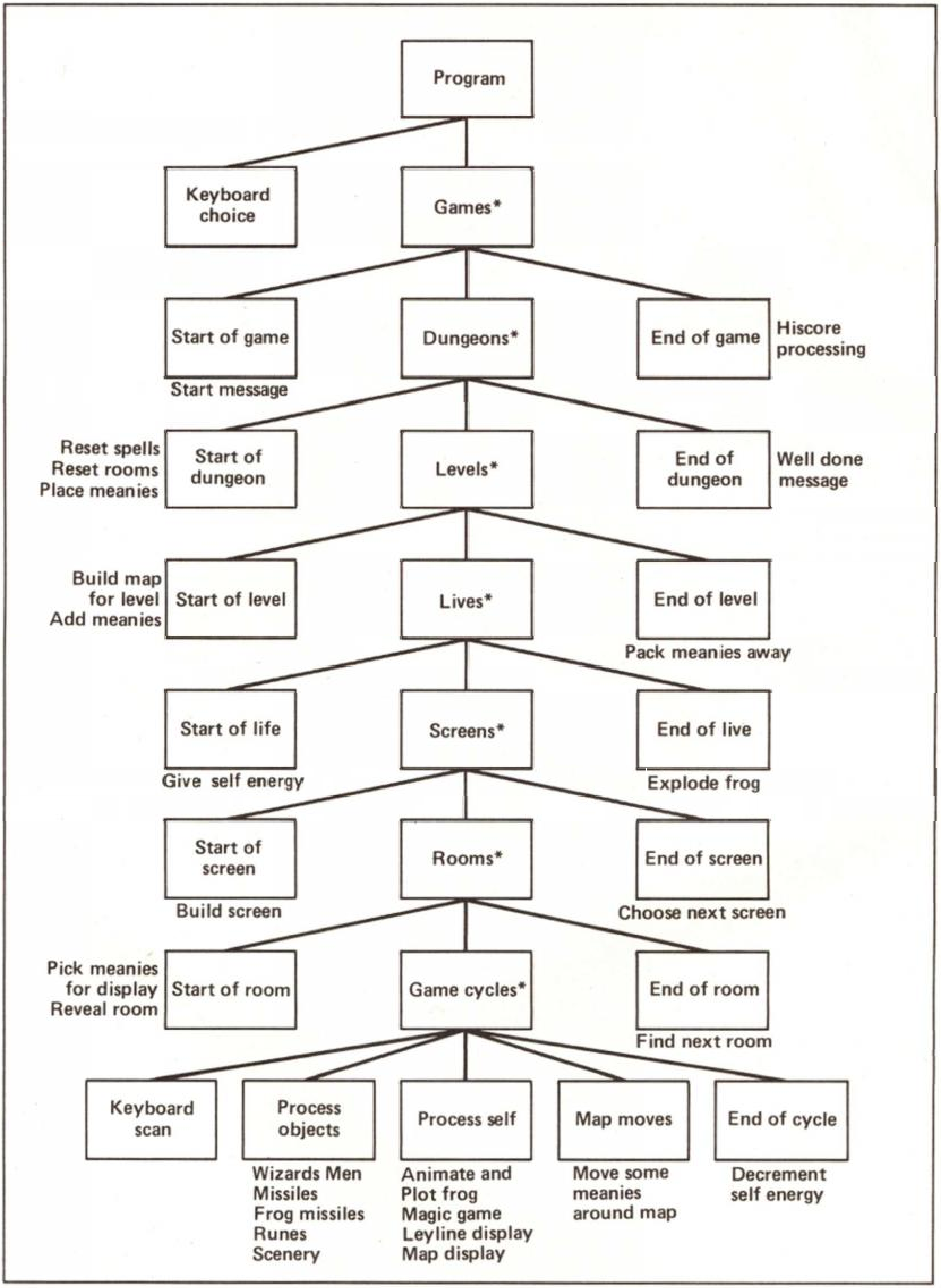


Figure 3: Ranarama Structure

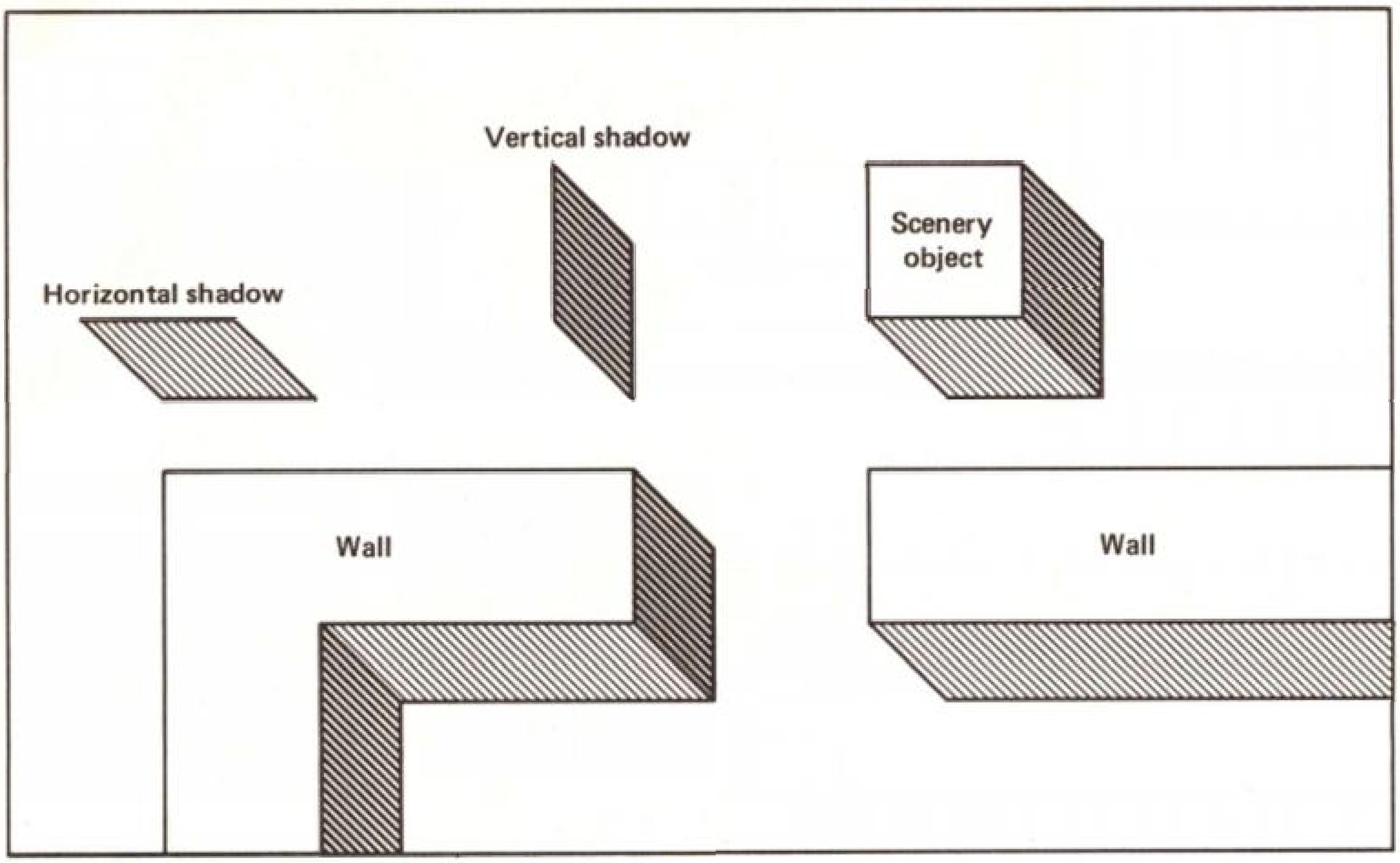


Figure 2: The Shadow processor

Hewson programmer, Andrew Braybrook, suggested the main character be a frog, as it fitted the sprite shape better. I tried it out and it worked. After a few hours of frantic coding it was jumping across the screen. I saw that Rana was the Latin for frog in the book I used for pictures of frogs.

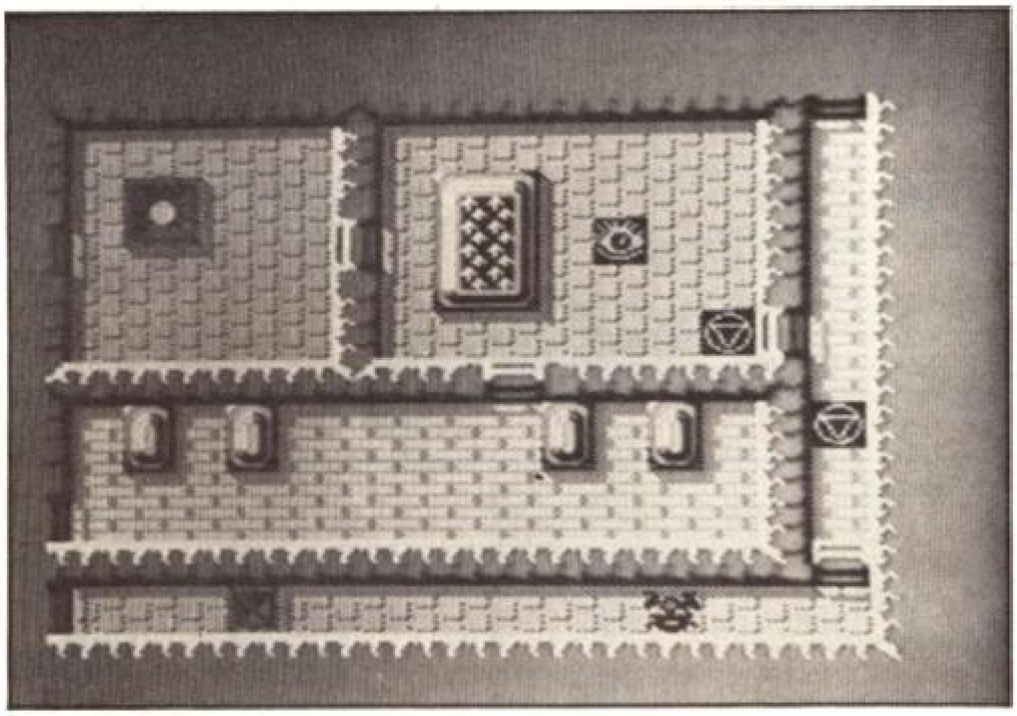
This gave me the name Ranarama. It was just the sort of tongue in cheek, but catchy name I was after.

Mid-project panics

There comes a time when you still have a mountain of work to complete, but have done so much that there is no turning back. I was at this stage when people observing my game said they did not like the magic combat part.

Originally it involved firing magic missiles across the screen at the opponent wizard. Missiles could collide with stationary missiles, splitting up into more missiles. It was possible to set up chain reactions, so the opponent had loads of missiles to dodge. The result was very pretty but I had a lot of trouble in designing graphics to go with the game. It was shelved for a month while I played with various ideas.

In the early hours of one morning while trying to incorporate the title in huge letters in the magic game, I had a brainwave; scrap a month's work and replace the magic game with a simple idea. The enemy wizard would mix up the title letters and the player would have to rearrange them before he was drained of energy. I tried the game using bits of paper and liked it. Two days later and it was designed, coded and running. The name Ranarama with four letter A's was ideal as it could never be too mixed up.



Ranarama: screenshot

Machine problems

In the latter stages of testing, machine crashes and disk corruptions crept up to an unacceptable level. I was regularly losing a few hours work and at one point had little chance of assembling the eight modules without a machine error or an assembler bug. (My OCP assembler regularly corrupts the label table. When you re-boot the machine and start again the problem disappears).

I traced the machine problem to the cheap and nasty connector. The trouble is keying on the machine gradually vibrates the connections until they lose contact despite much sticky tape and elastic bands. I thought a remote keyboard would be the answer and sent for the Saga 2001 only to find that this has to use the over burdened rear connector. It only has a blind interface (why didn't any reviewers mention this shortcoming).

To connect a printer a splitter cable is required. This contains

no less than four connections for data to bypass on its way from keyboard to CPU, and what mess the back of my machine was already, with a tangle of ribbon cables and a Beta disc interface. It never did work and I lost a weeks work trouble-shooting.

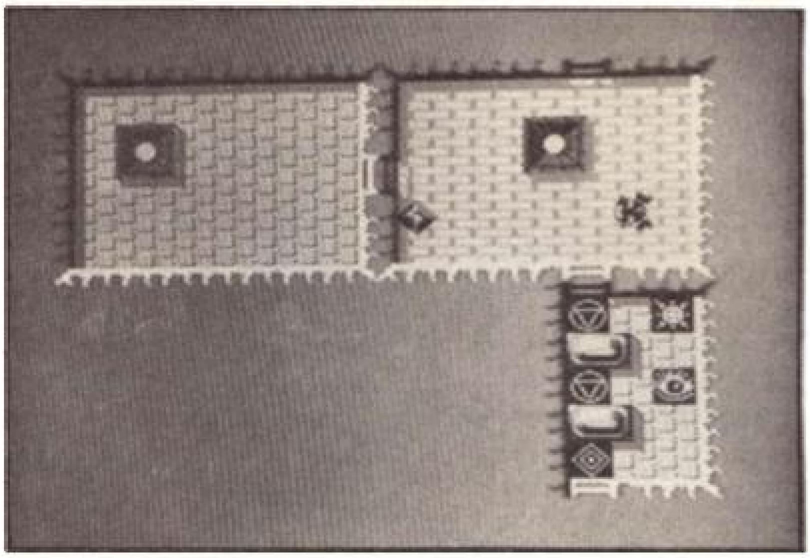
The problem was Z80 code running in low memory was sometimes corrupted. A LD L,1 was executed as LD A,1. I tried this over and over to prove it and was really sad to learn that no one could help, despite friendly support from Saga, and I finally gave in and sent the superb keyboard back.

This sort of problem just shows you must never rule out the unexpected. Whoever would expect that CPU to misinterpret an instruction. This problem solved, (see **Figure 4**) testing was underway again.

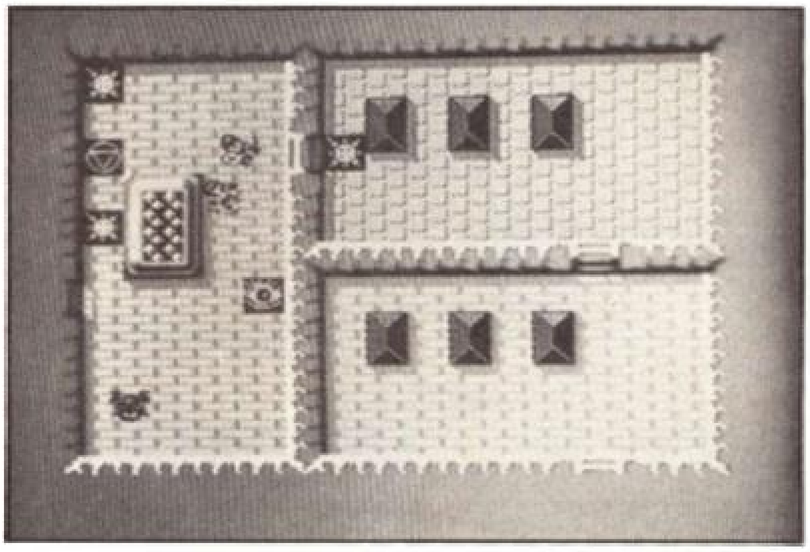
Play testing

At the time of writing the game has only been played by two people. In a few days the local computer club will have a go at it. You can expect to see it sometime early next year.

Perhaps these articles will have given you a greater insight to the work that goes into a game. But most of all I hope you have a really good zap.



Ranarama: screenshots



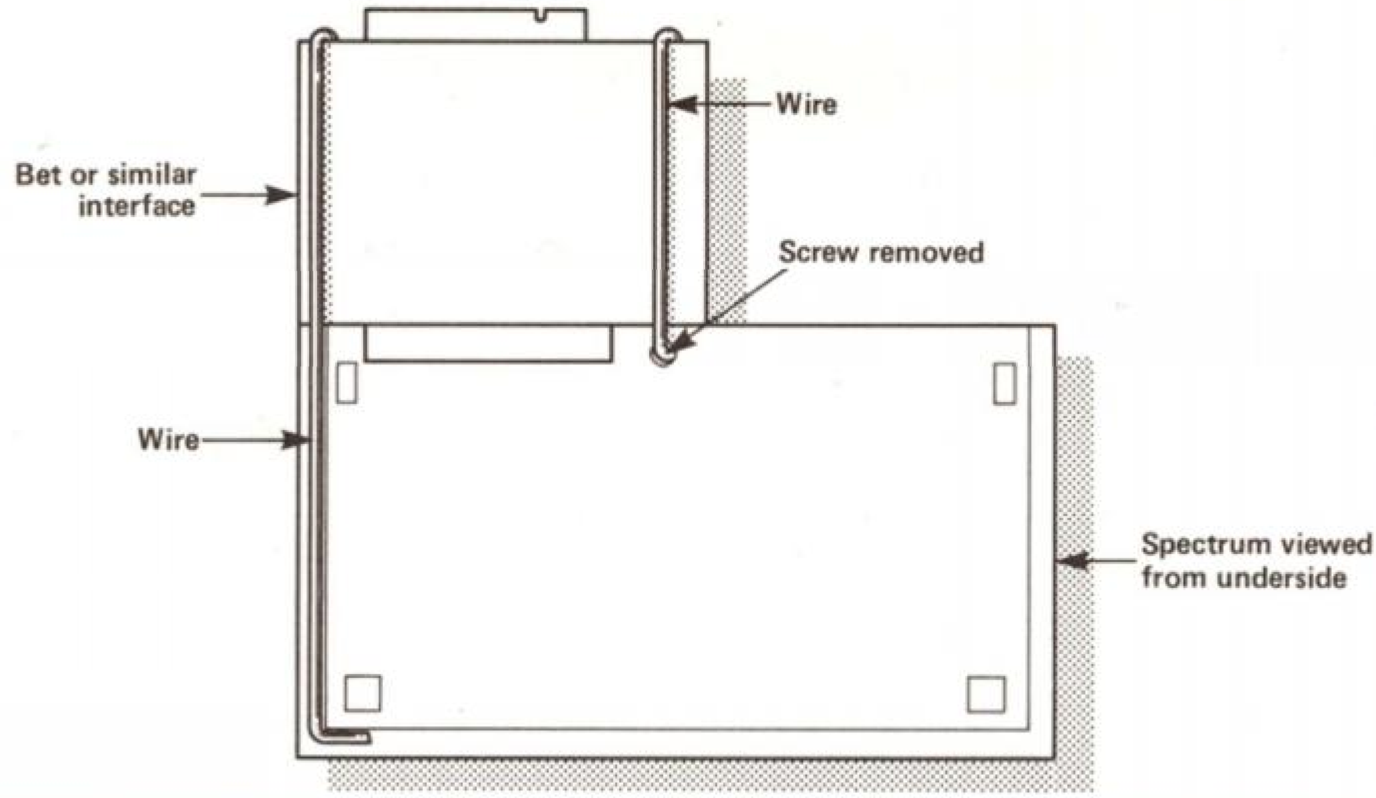
Professional improvisation

This is how I finally solved my machine crashing problem:

How to interface a Beta disc system with two pieces of coat hanger wire.

1. Remove screw from back edge of Spectrum.
2. Thoroughly clean connector at back with tape cleaning fluid.
3. Tighten connector clips by gently pushing each one with a screwdriver to bend them (if they are old and loose).
4. Cut two bits of a wire coat hanger and shape as in the diagram. The piece at the edge fits along the recess so the Spectrum can still sit on its feet.
5. Place the whole assembly on a board so when moved no pressure is exerted on the rear connector.

Figure 4



CROSSWIRES

Help and advice on a variety of technical topics from Ray Elder.

Timex/Tasword

First a plea from a reader in the RAF in Laarbruch.

Dear Sir,

Q *My dad who lives in Spain has a 48K Spectrum and a Timex Sinclair 2040 printer. Unfortunately he cannot get the printer to work with Tasword 2. Can you help?*

A The 2040 is the American equivalent of the Alphacom and it should operate without any problem with Tasword 2 providing it works on its own using the built-in LLIST, LPRINT and COPY commands.

If it does then the most likely problem is that he trying to use the centronics, full size printer routine instead of the Sinclair one.

Using C mode then key P should start the beast working, if not then it is possible that there is a fault in his copy of TAS 2.

Kempston Mouse/Art Studio

Finally a letter from London:

Dear Sir,

Q *I have a copy of Art Studio along with the Kempston mouse.*

On reading a few reviews about the Art Studio it appears that some other mice are supplied with a mouse software utility which enables the user to create mouse driven software of the Art Studio kind.

I would be interested to know if any software is available for the Kempston mouse, or if the existing software utility for the AMX mouse could be converted for the Kempston mouse by patching in appropriate port reading machine code.

I would be very grateful for any information you could give me on this subject.

Simon Atherley.

A AMX were helpful but had no idea if it would work, they said it should, but then again, as their interface is designed to emulate the BBC user port, perhaps it won't.

Anyway it could work or it could be a (relatively) simple job of patching the software or it may be impossible.

To start with AMX have supplied us with a copy of their software and I am sending it to Simon to try out on condition that he lets us know if it works. If it doesn't then he might like to send us any info he has from Kempston on their I/F and mouse so we can determine whether the AMX program can be patched.

Of course in the final instance the two may be completely incompatible, but at least we'll know for other users!

A final thanks to Tom Prosser who sent in a list of operating commands for a reader who purchased a disk drive interface secondhand and couldn't use it due to lack of instructions and who appealed through this column for info.

We have sent them on to him and are checking out Tom's own question in regard to his new Opus drive. We'll be in touch . . .

Printer hints

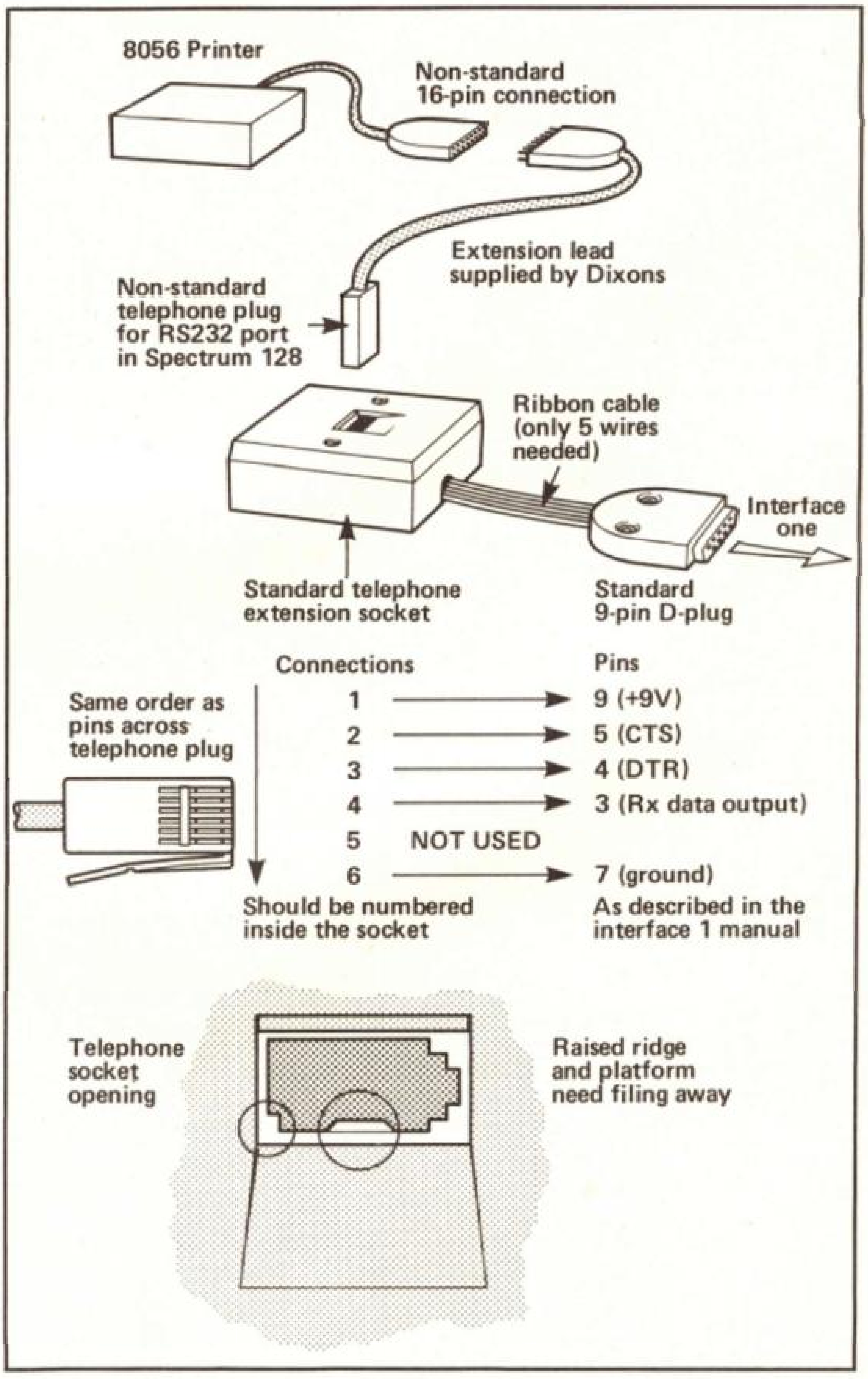
Alan H. Wadsworth of Herts writes with a bit of help and useful advice concerning the Dixons Serial 8056 printer that causes so much heartache to readers. We only received this just before our copy date and have not had time to build a mock up ourselves and so we cannot take any responsibility for it not operating. However, it seems sound in theory.

To use the printer with the Interface 1 RTS232 port you need an extension lead which is not (as far as we know) available commercially. Such a lead can be assembled for under £5.00 and can be easily constructed by an electronics engineer if you are not up to it yourself.

Construction is as shown in the diagram except the telephone socket, due to the fact that the RS232 socket is non-standard.

Two bits of the plastic inside the socket need to be filed away (an emery board cut to size and pressed with a small screwdriver is ideal) until it fits.

This will enable the printer to operate it in either 128 or 48K modes but it does mean purchasing an Interface 1.



SPECWORD 48/128

(Part 1)

A menu driven word processor for both 48 and 128 Spectrums by Stuart Nicholls.

This is a word processor that will meet the needs of most home users. Over three parts Specword will build up into a word processing program with many advanced features. Part 3 contains a complete operating manual.

Features

Specword has insert/overwrite modes, wordwrap, find, word count, block delete/move/copy/print, redefinable centronics control codes (30), 16K text space (blank lines only occupy 1 character space), lines of ANY length from one character to the full text length, margins, thirty one setable TABS, supports both the ZX printer and most centronics printers (with built in software for the Kempston interface) and RAM! file facilities for the 128K Spectrum.

Entering and Saving Specword

Specword 48/+/128K can be entered easily using the following procedure.

1. 128K Spectrum owner should type in the BASIC LISTING AS LIST 1. 48/K Spectrum owners will have to leave out the lines containing the instructions for RAM FILE loading and saving etc. ie lines 2220, 2410, 3620, 6510 & 6550. The program will work on their computers without these lines as the RAM FILE options are not available to them.
2. SAVE the BASIC program using SAVE "WP48/+/128"
3. Using the HEXLOADER/HEXDUMP dual purpose program (to be published in part 2) enter the machine code as HEXDUMP LIST 2.

REMEMBER TO RESET RAMTOP TO START ADDRESS−1 BEFORE RUNNING THIS HEXLOADER (ie ENTER CLEAR 33535 AS A DIRECT COMMAND FOR LIST 2).

The last byte in each line is the checksum byte, this being the sum of the previous 8 bytes modulo 256. The code should be entered one LINE at a time as a block of 18 characters (8 bytes + checksum) WITH NO SPACES BETWEEN THE CHARACTERS. To end the HEXLOADER program enter ZZ after the checksum byte. This can be done after any line in the dump and a printout of the number of bytes entered will be given (allowing you to save the block 'so far') and the last address. You will then be able to continue entering code from

Basic Listing: List 1

```
*** BASIC LISTING : LIST 1 ***
  15 LET c$=CHR$ 19+CHR$ 1+CHR$ 20+CHR$ 1: LET p$=CHR$ 22+CHR$ 21+CHR$ o+c$
  16 LET ld=2000: LET sv=3000
  17 LET rd=8500: LET pt=4000
  18 LET sm=8000
  20 BORDER 7: CLS : POKE 23607,175: LET e=13: LET s=5: LET a=USR 40704
  22 PRINT  BRIGHT 1;AT 13,7;"   ERASE!        "
  23 GO SUB rd
  25 IF x=13 THEN  GO TO 6500
  30 GO TO (x-5)*1000
1000 CLS : LET e=7: LET s=5: LET a=USR 40961: GO SUB rd
1002 IF x=7 THEN  GO TO menu
1010 GO SUB 8400
1020 IF NOT n THEN  GO TO 1000
1030 LET a=USR 33536
1035 IF a=1 THEN  GO TO 9100
1040 IF a<>3 THEN  GO TO menu
1050 LET a=USR 39873
1060 INPUT  LINE x$
1070 IF x$= "" THEN  LET x$=y$
1080 LET y$=x$: LET a=USR 33551: GO TO 1040
2000 CLS : LET e=8: LET s=5: LET a=USR 41043: GO SUB rd
2200 IF x=8 THEN  GO TO menu
2205 IF x=6 THEN  CLS : GO TO 2225
2210 IF NOT spec THEN  GO SUB 7470: GO TO ld
2220 CLS : CAT !
2225 PRINT p$;" Title for Loading ?      ";: GO SUB sm
2230 INPUT  LINE d$: IF d$="" AND x=6 THEN  GO TO 2275
2240 IF LEN d$>10 THEN  GO SUB 7400: GO TO ld
2250 IF d$="" THEN  GO SUB 7440: GO TO ld
2260 FOR a=1 TO LEN d$: IF d$(a)<" " THEN  GO SUB 7420: GO TO ld
2270 IF d$(a)>CHR$ 127 THEN  GO SUB 7420: GO TO ld
2275 LET a=11-LEN d$
2276 IF x=7 THEN  GO TO 2400
2277 PRINT AT 20,o;" Play Tape....."
2280 PRINT p$;" Loading ";CHR$ 34;d$;CHR$ 34;s$( TO a);"    ";: GO SUB sm
2290 LOAD d$CODE 49152: IF d$<>"" THEN  GO TO menu
2300 FOR a=49203 TO 49212: LET b= PEEK a: IF b=255 THEN  GO TO menu
2310 LET d$=d$+CHR$ b: NEXT a
2320 GO TO menu
2400 PRINT p$;" Loading ";CHR$ 34;d$;CHR$ 34;s$( TO a);"    ";: GO SUB sm
2410 LOAD ! d$CODE 49152
2420 PAUSE 100: GO TO menu
```

where you left off.

The HEXLOADER program will error trap ANY incorrect line inputs and prompt you to re-enter wrongly keyed in lines.
4. SAVE this block of code as "code1" CODE 33536,9407
5. Using the same HEXLOADER enter the machine code as the HEXDUMP LIST 3.
6. SAVE this block of code as "code2" CODE 44800,2519
7. When you are happy that all has been saved and verified then RESET your Spectrum and enter CLEAR 32767 as a direct command.
8. LOAD the BASIC program.
9. LOAD the CODE code1.
10. LOAD the CODE code2.
11. FINALLY (if you have managed to get this far without our fingers dropping off), SAVE the complete program using: SAVE "WP48/+/128" LINE 9800:SAVE "specode128" CODE 32768,14600.
12. Should any of the functions not work then the machine code can be checked using the HEXLOADER/HEXDUMP dual purpose program. The HEXDUMP option will print out to the screen or printer any block of memory in the same form as the HEXDUMP LISTS 2 and 3 (The easiest way to check the code is to compare checksum bytes).

```
3000 CLS : LET e=10: LET s=5: LET a=USR 41144: GO SUB rd
3200 IF x=10 THEN  GO TO menu
3210 IF x=8 THEN  GO SUB 8400: GO TO sv
3220 IF x=9 THEN  GO TO 9850
3400 IF x<>6 THEN  GO TO 3600
3402 IF d$="" THEN  GO SUB 7440: GO TO sv
3405 LET a= 11-LEN d$
3410 CLS : PRINT p$;" Saving ";CHR$ 34;d$;CHR$ 34;s$( TO a);"      ";: GO SUB sm
3420 SAVE d$CODE 49152,(PEEK 49160+256*PEEK 49161)-49151
3430 PRINT p$;" Verify ? <Y><N>              ";: GO SUB sm
3440 LET a$=INKEY$: IF a$<>"y" AND a$<>"Y" AND a$<>"n" AND a$<>"N" THEN  GO TO 3
440
3450 IF a$="n" OR a$="N" THEN  GO TO sv
3460 PRINT p$;" Rewind Tape and play      ";: PRINT AT 18,0;: VERIFY d$CODE
3470 LET m=USR 42302: GO SUB 7480: GO TO sv
3600 IF NOT spec THEN  GO SUB 7470: GO TO sv
3603 IF d$="" THEN  GO SUB 7440: GO TO sv
3605 LET a= 11-LEN d$
3610 CLS : PRINT p$;" Saving ";CHR$ 34;d$;CHR$ 34;s$( TO a);"      ";: GO SUB sm
3620 SAVE ! d$CODE 49152,(PEEK 49160+256*PEEK 49161)-49151
3630 PAUSE 100: GO TO sv
4000 CLS : LET e=11: LET s=5: LET a=USR 41283: GO SUB rd
4200 RESTORE 4210: FOR a=i TO 6: READ y,z: IF x=y THEN  GO TO z
4205 NEXT a
4210 DATA 6,4300,7,4300,8,4800,9,4500,10,4700,11,20
4300 CLS : LET m=USR 42347: GO SUB sm
4305 PAUSE o: PRINT p$;s$( TO 25);: IF x=6 THEN  GO TO 4320
4310 LET bk=USR 33557: IF bk=65535 THEN  GO TO 4000
4315 GO TO 4330
4320 LET a=USR 33554
4330 LET p=page: LET e=o
4340 PRINT AT 20,o;"Page ";p
4350 IF top THEN  LPRINT s$( TO mar);"(";p;")"
4355 IF p=page AND x=7 THEN  LPRINT s$( TO bk);
4360 FOR y=i TO (lpp-top-bot): IF e THEN  LPRINT : GO TO 4380
4370 LPRINT s$( TO mar);: LET a=USR 33545: IF a<>i THEN  LET e=i
4380 FOR w=i TO sp: LPRINT : NEXT w
4390 NEXT y
4400 IF bot THEN  LPRINT s$( TO mar);"(";p;")"
4410 IF auto THEN  LPRINT CHR$ 12
4420 IF NOT wait THEN  GO TO 4440
4430 LET m=USR 42392: PAUSE o: PRINT p$;s$( TO 24);
4440 LET p=p+i: IF NOT e THEN  GO TO 4340
4450 IF x=7 THEN  LPRINT
4460 GO TO pt
4500 CLS : LET s=2: LET e=10: LET a=USR 41441: GO SUB rd
4510 IF x=10 THEN  GO TO pt
4515 LET k$="BGHIJKLMNOPTUVY"
4520 LET l$=k$(x-2)
4530 CLS : LET s=5: LET e=8: LET a=USR 41789: GO SUB rd
4540 IF x=8 THEN  GO TO pt
4545 IF x=7 THEN  LET l$=CHR$ ((CODE l$)+32)
4550 LET j$="TtYyIiOoPpGgHhJjKkLlVvBbNnMmUu"
4555 FOR a=i TO 30: IF j$(a)=l$ THEN  GO TO 4565
4560 NEXT a
4565 LET c=(a-i)*5+33776
4570 CLS : LET x=6: LET y=7: LET z=16
4575 PRINT AT 5,y;c$;s$( TO z): FOR a=6 TO 10: PRINT  BRIGHT i;AT a,y;s$( TO z):
 NEXT a
4580 PRINT AT 5,8;c$;l$;AT 5,17;: GO SUB sm: FOR a=6 TO 9: PRINT  BRIGHT i;AT a,
11;PEEK (c+(a-4));"      ": NEXT a: PRINT  BRIGHT i;AT 10,11;"Exit"
4585 PLOT 56,136: DRAW 128,o: DRAW o,-49: DRAW -128,o: DRAW o,49
4590 GO SUB 7990
4600 PRINT  PAPER 5; BRIGHT i; OVER i;AT x,y;s$( TO z)
4605 LET a$=INKEY$: IF a$=CHR$ 13 AND x=10 THEN  GO TO 4500
4611 IF a$=CHR$ 11 OR a$=CHR$ 10 THEN  GO TO 4670
4612 IF a$<"0" OR a$>"9" THEN  GO TO 4605
4614 LET w=18: GO SUB 7300: IF b$="" THEN  GO TO 4600
4630 IF a>255 THEN  PRINT  BRIGHT i;AT x,w;"   ": GO TO 4600
4640 POKE c+x-4,a: GO TO 4580
4670 GO SUB 7200: GO TO 4600
4700 LET a=USR 33542: LET a=USR 42164: FOR a=i TO 300: NEXT a: GO TO pt
4800 CLS : LET x=4
4802 LET a=USR 40370
4804 GO SUB 7990
4807 PLOT 15,152: DRAW 225,o: DRAW o,-81: DRAW -225,o: DRAW o,81
4820 PRINT  BRIGHT i;AT 4,22;"(";lpp;")  ";AT 5,22;"(";sp;")  ";AT 6,22;"(";mar;
")  ";AT 7,22;"(";page;")  ";AT 8,22;"(";CHR$ (78+(11*(top=i)));")";AT 9,22;"(";
CHR$ (78+(11*(bot=i)));")";AT 10,22;"(";CHR$ (78+(11*(wait=i)));")";AT 11,22;"("
;CHR$ (78+(11*(auto=i)));")"
4835 PRINT  PAPER 5; BRIGHT i; OVER i;AT x,2;s$( TO 28)
4840 LET a$=INKEY$
4855 IF a$=CHR$ 13 AND x=12 THEN  GO TO pt
4860 IF a$="y" OR a$="Y" OR a$="n" OR a$="N" THEN  GO TO 4900
4870 IF a$>"/" AND a$<":" THEN  GO TO 4950
4875 IF a$<>CHR$ 10 AND a$<>CHR$ 11 THEN  GO TO 4840
4877 PRINT  BRIGHT i; OVER i;AT x,2;s$( TO 28)
4880 LET x=x+(a$=CHR$ 10)-(a$=CHR$ 11)
4885 IF x>12 THEN  LET x=4
4890 IF x<4 THEN  LET x=12
```

```
4895 GO TO 4835
4900 IF x<8 THEN  GO TO 4835
4902 LET a= (CODE a$-32)
4905 IF x=8 THEN  LET top=o+(a=89)
4910 IF x=9 THEN  LET bot=o+(a=89)
4915 IF x=10 THEN  LET wait=o+(a=89)
4920 IF x=11 THEN  LET auto=o+(a=89)
4925 GO TO 4820
4950 IF x>7 THEN  GO TO 4835
4955 LET w=27: GO SUB 7300: IF a$=CHR$ 12 THEN  GO TO 4820
4976 IF x=4 THEN  LET lpp=a
4978 IF x=5 THEN  LET sp=a
4980 IF x=6 THEN  LET mar=a
4982 IF x=7 THEN  LET page=a
4984 PRINT AT x,27;"   ": GO TO 4820
5000 IF d$="" THEN  GO SUB 7440: GO TO menu
5005 IF PEEK 49152=o THEN  GO SUB 7450: GO TO menu
5010 PAUSE 100: LET a=USR 33539: GO TO 1034
6000 IF NOT spec THEN  GO SUB 7470: GO TO menu
6005 CLS : LET a=USR 42074: GO SUB 8000: PRINT
6010 POKE 23692,255: CAT !: PRINT
6020 LET a=USR 42120: GO SUB 8000
6030 PAUSE o: GO TO menu
6500 IF NOT spec THEN  GO TO 6000
6505 CLS : PRINT c$;" List of RAM! files :";: GO SUB sm
6510 PRINT : CAT !
6520 PRINT p$;" ERASE! name ?:<ENTER=skip";: GO SUB sm
6530 INPUT  LINE n$
6540 IF n$="" THEN  GO TO menu
6550 ERASE ! n$
6560 GO TO menu
7000 CLS : LET x=6: LET y=5: LET z=22
7010 GO SUB 7990
7020 LET a=USR 40197
7030 PLOT 39,136: DRAW 177,o: DRAW o,-49: DRAW -177,o: DRAW o,49
7035 PRINT  BRIGHT i;AT 6,20;tc;"  ";AT 7,20;wc;"  ";AT 8,20;bc;"  ";AT 9,20;pc;
"  "
7040 PRINT  PAPER 5; BRIGHT i; OVER i;AT x,y;s$( TO z)
7050 PAUSE o: LET a$=INKEY$: IF a$=CHR$ 13 AND x=10 THEN  GO TO menu
7060 IF a$=CHR$ 11 OR a$=CHR$ 10 THEN  GO SUB 7200: GO TO 7040
7070 IF a$<"0" OR a$>"9" THEN  GO TO 7050
7080 LET w=24: GO SUB 7300: IF b$="" THEN  GO TO 7040
7090 IF a>255 THEN  PRINT  BRIGHT i;AT x,w;"   ": GO TO 7040
7100 IF x=6 THEN  LET tc=a: POKE 49175,a: POKE 33718,a
7110 IF x=7 THEN  LET wc=a: POKE 49176,a: POKE 33719,a
7120 IF x=8 THEN  LET bc=a: POKE 49177,a: POKE 33720,a
7125 IF x=9 THEN  LET pc=a: POKE 36083,a
7130 PRINT AT x,w;"   ";: GO TO 7035
7200 PRINT  BRIGHT i; OVER i;AT x,y;s$( TO z)
7210 LET x=x+(a$=CHR$ 10)-(a$=CHR$ 11): IF x=11 THEN  LET x=6
7220 IF x=5 THEN  LET x=10
7230 RETURN
7300 LET a=i: LET b$=""
7310 LET b$=b$+a$: PRINT  BRIGHT i; PAPER 2;AT x,w;b$
7320 PAUSE 0: LET a$=INKEY$: IF a$=CHR$ 13 THEN  GO TO 7360
7330 IF a$=CHR$ 12 THEN  PRINT  BRIGHT i;AT x,w;"   ": LET b$="": RETURN
7340 IF a$<"0" OR a$>"9" THEN  GO TO 7320
7350 LET a=a+i: IF a<>4 THEN  GO TO 7310
7360 LET a=VAL b$: RETURN
7400 LET a=USR 41890: GO TO 7480
7420 LET a=USR 41936: GO TO 7480
7440 LET a=USR 41982: GO TO 7480
7450 LET a=USR 42648: GO TO 7480
7470 LET a=USR 42437
7480 FOR a=i TO 200: NEXT a: RETURN
7990 IF spec THEN  LET a=USR 40705: RETURN
7995 LET a=USR 42026
8000 LET a=USR 40758: RETURN
8400 CLS : LET n=o: LET a=USR 42483: GO SUB sm: INPUT  LINE d$
8420 IF LEN d$>10 THEN  GO TO 7400
8430 IF d$="" THEN  GO TO 7440
8440 FOR a=i TO LEN d$: IF d$(a)<" " OR d$(a)>CHR$ 127 THEN  GO TO 7420
8470 FOR a=33747 TO 33756: POKE a,255: NEXT a
8475 FOR a=49203 TO 49212: POKE a,255: NEXT a
8480 LET b=i: FOR a=33747 TO 33747+LEN d$-i: POKE a,CODE d$(b): LET b=b+i: NEXT
a
8485 LET b=i: FOR a=49203 TO 49203+LEN d$-i: POKE a,CODE d$(b): LET b=b+i: NEXT
a
8490 PRINT p$;" Title ";CHR$ 34;d$;CHR$ 34;s$( TO 11-LEN d$);"       ";: GO SUB s
m: PAUSE 100: LET n=i: RETURN
8500 PLOT 56,176-(s*8): DRAW 128,o: DRAW o,-(e-s+i)*8-i: DRAW -128,o: DRAW o,i+(
e-s+i)*8
8520 GO SUB 7990
8530 LET x=s+i
8540 PRINT  PAPER 5; BRIGHT i; OVER i;AT x,7;s$( TO 16)
8550 LET a$=INKEY$: IF a$=CHR$ 13 THEN  RETURN
8560 IF a$<>CHR$ 10 AND a$<>CHR$ 11 THEN  GO TO 8550
8570 PRINT  OVER i; BRIGHT i;AT x,7;s$( TO 16)
8580 LET x=x+(a$=CHR$ 10)-(a$=CHR$ 11)
8590 IF x=s THEN  LET x=e
8595 IF x=e+i THEN  LET x=s+i
8597 GO TO 8540
9800 CLEAR 32767: LOAD "specode128"CODE : CLS : LET a=USR 42529
9805 LET i=1: LET o=0
9810 LET y$=" ": LET d$="": LET auto=o: LET page=i: LET top=o: LET bot=0: LET lp
p=61: LET mar=o: LET sp=o: LET wait=i: LET menu=20: LET s$="
         "
9820 LET tc=PEEK 33718: LET wc=PEEK 33719: LET bc=PEEK 33720: LET pc=PEEK 36083:
 LET spec=USR 40767: GO TO 15
9850 PRINT p$;" Saving ";CHR$ 34;"WP48/+/128";CHR$ 34;"     "
9860 SAVE "WP48/+/128" LINE 9800
9870 PRINT p$;" Saving ";CHR$ 34;"specode128";CHR$ 34;"     "
9900 SAVE "specode128"CODE 32768,14600
9910 PRINT p$;" Verify Y/N ?             "
9920 PAUSE 0: IF INKEY$="n" OR INKEY$="N" THEN  GO TO sv
9925 IF INKEY$<>"Y" AND INKEY$<>"y" THEN  GO TO 9920
9930 PRINT p$;" Rewind TAPE and play     "
9940 PRINT AT 15,0;: VERIFY "WP48/+/128"
9950 PRINT AT 17,0;: VERIFY "specode128"CODE
9960 LET a=USR 42302: GO SUB 7480: GO TO sv
```

READ ONLY

Books on C and Sir C in the return of the back page.

C for Beginners
Melbourne House Publishers
£10.95

With the popularity of 'C' as the favorite alternative to BASIC for programming (see ZXC, August 1986, page 28) it comes as no surprise that there have been several books published recently on 'C'. That prolific author, Ian Sinclair, joins the crowd with his offering, 'C for Beginners'.

Sinclair's concept of a beginner is someone who knows a little BASIC, but now wants to try something else. If that fits you, then this book can be highly recommended.

The opening of the book is fairly standard; what are high and low level languages? — why do we need so many languages? — compilers and interpreters — and one important difference between BASIC and 'C', structured programming.

After that, you get the fundamentals of 'C' carefully and clearly explained. You have to wait for page 48 of this book of 228 pages to get the first very simple 'C' routine to try out. But, by that stage, the reader should be well in tune with the concepts of 'C'.

From that point onwards, Sinclair carefully builds up the reader's knowledge; by the end anyone should feel reasonably comfortable with the language, ready to proceed to a more advanced book.

Routines in the book were written for the Hisoft 'C' compiler for the Amstrad; these seem to work equally well on Hisoft's Spectrum 'C' compiler. 'C' is meant to be a portable language, so most machine/compiler combinations should accept the fairly simple example routines given in the text.

There are many comparisons made throughout the book to BASIC, so the BASIC programmer should experience no major difficulties. At the back of the book, there is a useful short reference section giving the 'C' equivalent of many BASIC keywords.

Most books on 'C' are a little over-priced, so, by comparison, £10.95 for this excellent little book for beginners represents a good buy.

David Nowotnik

Sinclair and the 'Sunrise' Technology
Penguin Books Ltd.
£3.95

The authors, Ian Adamson and Richard Kennedy, have varied and colourful backgrounds including a few credits in the field of computer journalism, though noticeably not in the Sinclair field until now, at least according to their given personal briefs.

This very 'distance' from the whole subject probably stood them in good stead in the writing of the book and contributes to its lack of emotional colouring in favour of the charismatic Sir C.

Unlike some tomes which I have read, the authors do not view Sir Clive Sinclair through rose tinted glasses but by using facts, reports and documented statements they have researched and produced what must be the definitive account of his business ventures to date.

Devotees will find that their concept and perceptions of the man may well need a little re-appraising, but even so he has been treated gently and emerges as a likeable personality.

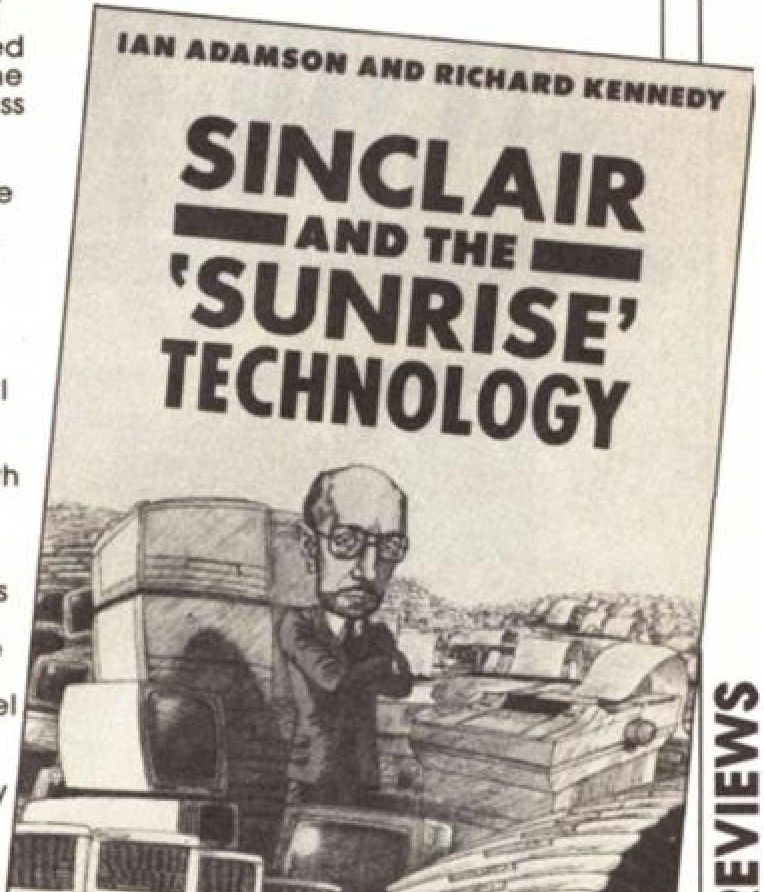
You may have gathered by now that this is not a superficial sop to the masses, it does not simply present a potted history of the events and pad it out with opinion, technical references, photographs and diagrams. Indeed there are none of the latter and the technical input is limited to brief and easily understood explanations where necessary.

Opinions are part and parcel of the text although they are presented in academically acceptable form as reasonably argued ideas which are well supported by documented evidence and the events of the time. Without taking months to check and correlate on the material presented by them I can only concede that they have a persuasive and almost certainly accurate account of the whole affair.

I am not sure of the market to which this book is aimed, as an authorative academic book it is valuable for schools and colleges for social studies, business and perhaps political courses. The rather dry, unsensational approach would tend to limit its appeal to the more thoughtful individual and in this category I would also put the established hobbyist (or 'obsessive' as the book occasionally labels us), and perhaps the anti-Sinclair brigade who seem to look for any excuse to put him down.

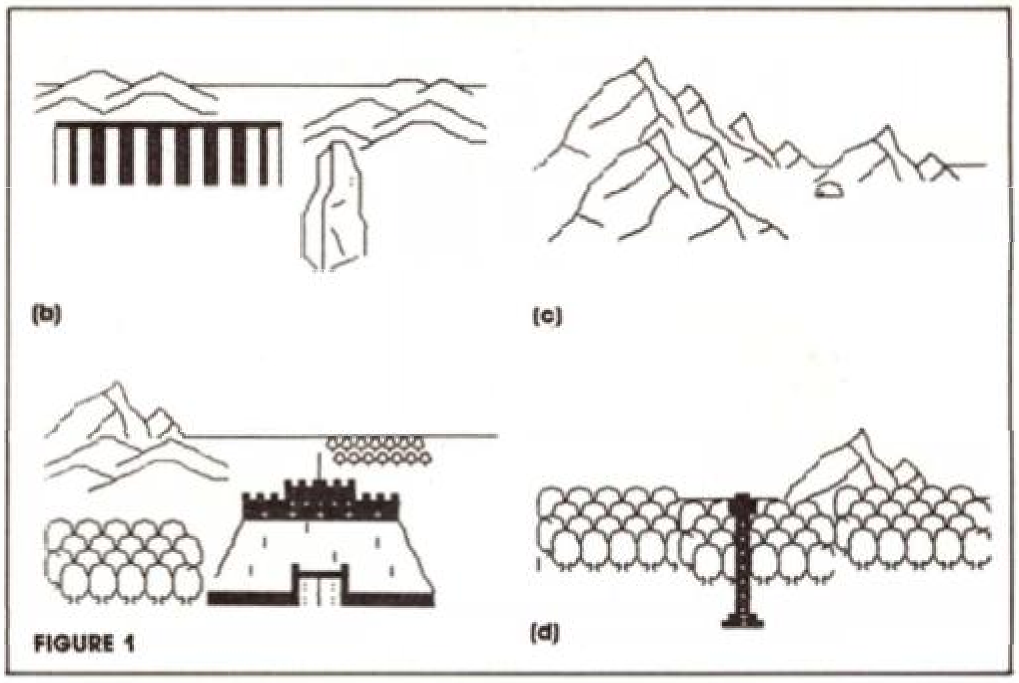
The great majority of Sinclair users will probably not be interested in this book though I found it a fascinating and plausible account of a period of history which may be regarded in our future as the turning point of a new era. Only time will tell . . .

Ray Elder



Anyone who has watched the ever-changing landscapes of Lords of Midnight must have a secret wish to have a similar facility in their own adventure programs. This graphics utility can go a long way to granting that wish. It can hold the data for a full screen, hi-res illustration in only 30 bytes, so with it in the top 32K of memory you can generate any of 1000 illustrations in less than one second.

This means that you could hve four view, looking north, south, east and west, of each location on a 25 x 10 grid. The screen copies shown in fig. 1 illustrate a range of the possible landscapes you could produce.



(b) (c) (d)

FIGURE 1

Landscapes

If you have looked closely at Midnight's landscapes you will have noticed that they are made up of a number of picture elements (trees, mountains etc) drawn to different sizes. My program uses the same idea but operates in Basic, making it easier for most people to handle. Look at Program 1. Lines 10-19 hold the graphics for the picture elements (let's call them picsubs). These include moutains, forest, hills, caverns, hedges, monoliths, lakes, towers, keeps and castles, and can be drawn in two sizes; lines 10-19 for distant views, and lines 20-29 for close views. Each can be PRINTed At a screen position v, h; this marking the top left corner of the element block. Line 9 sorts out what is to be printed where. The multidimensional array L$() holds this information in a series of strings each made up of five groups of three, two-digit numbers. The first pair in a triplet are more difficult to position. You're also not limited to five picsubs per landscape. You could use as many as you like by altering the loop limit value but you must use the same number each time, so remember — the more elements you use per picture the less number of pictures you can squeeze in. A careful examination of Midnight showed five to be about the right number but the choice is yours.

RANDOM MEMORY



is the row on screen, the second is the column, and the third is the program line number where the picsubsubroutine is located. The j loop operates five times producing the five picsubs which make up the completed landscape. Before these are pirnted INK and PAPER are both set to black to nothing appears until a short machine code routine is called which pokes all the attributes with 8, thus making the illustration appear as white on black.

Obviously, now you know the method you could go off and do your own thing. I've chosen to illustrate a Middle Earth type landscape but by changing the UDGs you could produce a futuristic, heroic, or just plain ordinary one. The dimensions of each picsub is also a matter of choice. Large ones would produce a "busier" screen but

UDGs

Assuming you're happy with my set-up, type in Program 1. This is the one you'll need to incorporate in your own adventure programs to run the utility. When you've typed it in RUN and enter the data given in Table A. This is the machine code and UDG data. Delete line 1, then SAVE the program and code with

SAVE "landscaper" LINE 9999: SAVE "landcode" CODE 65351, 185

and VERIFY. One very important point to remember: as you have deleted line 1 the variables used to replace numbers and save bytes are now only held in the variables area. All program using this utility must therefore autostart using LINE . . . when

they are SAVEd. They must also be restarted using GOTO, not RUN as this clears the variables area. You must, of course, not reuse the variable names used in Line 1 or the values would be corrupted.

By now a ghastly thought may have crossed your mind. You've got to a) think up 1000 landscapes and, b) work out the data for each and get it into the data array I$()! I can offer advice on the first problem and an Assembler/Editor to aid the second.

Let's leave the keying in of the Assembler 'til late and worry first about the design and layout of the illustrations.

Map

You first need a good idea of what our adventure map looks like overall. By way of example, fig. 2 gives a sample map fc- us to use. It's only 3 x 3 locations — anyone who is clever enough to start with the full 25 x 10 doesn't need my help! You will notice that each location is subdivided into four. This makes it easier to decide what is where, and what size for each landscape. The letters appearing in each quarter square symbolise the picsubs which are to appear. Table B gives a key for these.

Clyde Bish with a utility for producing 'Landscaped' graphics.

In each location they are deemed to be standing in the centre (where the dotted lines cross) and looking N. S. E. or W. So if you were in square 1,1 (where the dot is) and looking east, you would see a castle near right, and forest left, with another area of forest in the right distance and hills leading up to mountains in the left distance. This is the landscape shown in fig. 1 (a). It would be worth now trying to work out what the views for the rest of that square, or other squares would be to get the hang of the system. What happens at the edge of the map (e.g. square 1,1 looking N. or W.) you must decide. You could surround your lands with impassable mountains, or simply have a no-go featureless horizon.

OK, so you think you know what each landscape is to look like, but it would help to be able

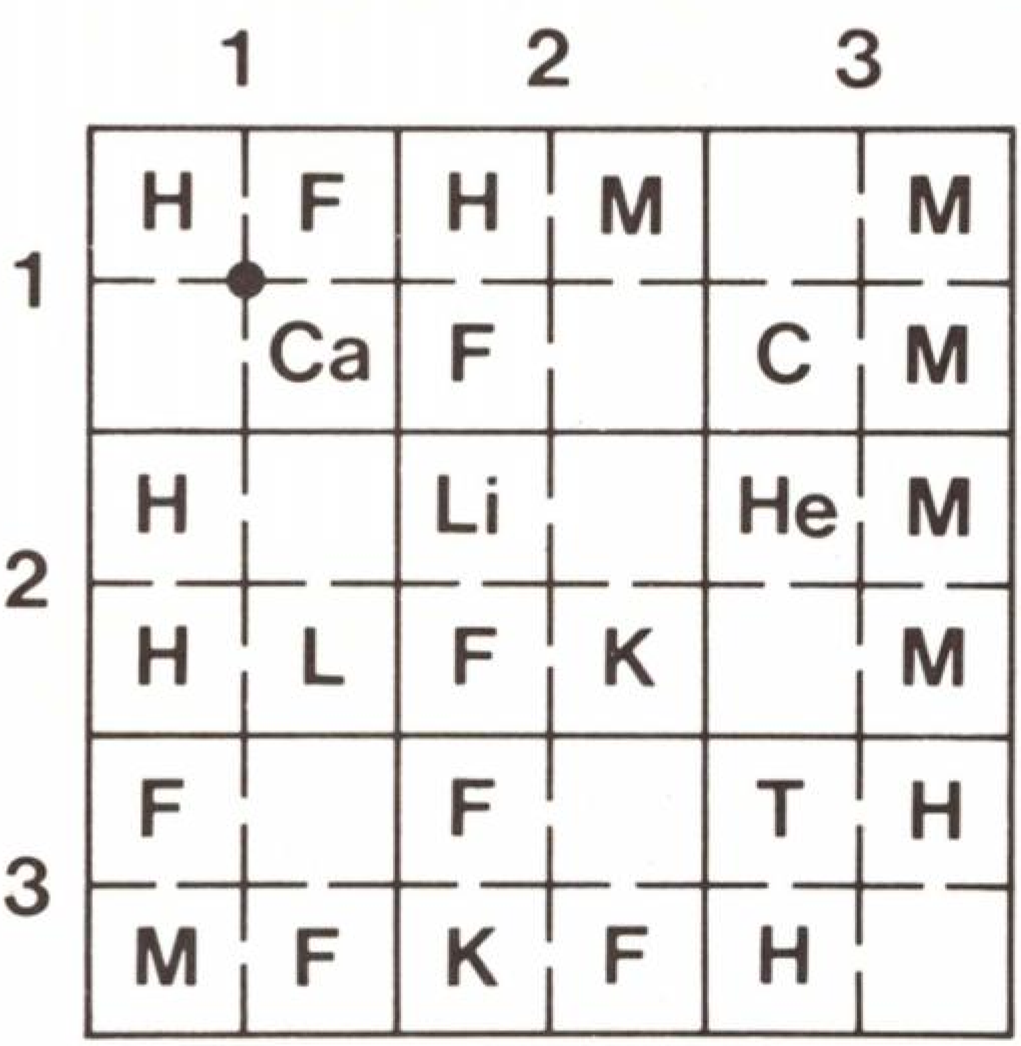


FIGURE 2: Sample Map

TABLE A

33	0	88	1	192	2	22	7
114	35	11	120	177	32	249	201
62	3	28	32	64	64	64	128
128	128	64	64	36	26	2	2
4	192	56	4	2	2	2	1
1	1	2	2	36	88	64	64
32	56	68	68	130	130	108	16
16	131	68	68	40	40	198	1
1	128	64	64	32	32	192	0
0	3	4	4	8	8	6	1
1	128	96	28	2	1	0	0
0	0	0	0	0	192	56	4
3	3	4	56	192	0	0	0
0	0	0	0	0	7	24	96
128	128	96	16	8	8	6	1
1	0	1	2	2	12	16	32
192	128	64	64	32	16	16	16
8	8	8	4	4	4	2	2
1	1	1	1	2	4	4	8
8	16	16	32	32	32	32	64
128	16	16	16	16	16	16	16
16	128	128	128	128	128	128	128
128	0	0	0	0	255	0	0
0							

to see each one on screen and make any necessary artistic adjustments. This is where the Assembler/Editor comes in. LOAD in Program 1 if it's not already in the machine and add to it Program 2. SAVE it to LINE 9999 plus the machine code/udg data as before.

When LOADed normally it will autostart at LINE 9999, then jump to the menu, but for the moment start it with GOTO (not RUN!) 900. This is also how you would restart the Assembler if you get an error message after a failed VERIFY perhaps.

The Menu lists the available options. Let's have a look at the facilities available. If you get into any option by mistake, pressing just ENTER will escape.

TABLE B

Key to map and picsub codes

	Symbol	Code Far	Code Near
Mountain	M	10	20
Forest	F	11	21
Hills	H	12	22
Cavern	C	13	23
Monolith	Li	14	24
Lake	L	15	25
Henge	He	16	26
Tower	T	17	27
Keep	K	18	28
Castle	Ca	19	29

Grid	View	Picel	Row	Column	Picel No.
1,1	1	1	7	0	12
		2	8	0	22
		3	12	2	12
		4	9	20	21
		5	12	17	21
	2	1	4	0	10
		2	7	0	22
		3	8	20	11
		4	9	11	29
		5	14	0	21
	3	1	7	25	12
		2	8	19	22
		3	9	12	15
		4	12	24	12
		5	10	1	29
	4	1	7	20	12
		2	8	26	12
		3	9	19	22
		4	15	15	12
		5	15	15	12

TABLE C

1. Assemble
This option displays each view as you build it up. The "width?" prompt is the number of grid squares across your map. (Enter "c" here if you are continuing with an unfinished landscape). Height is the vertical number of grid squares. For each picsub of each view enter row, column and picsub number (If you can't remember the latter, enter "h" for a help page). Remember the views go clockwise in the sequence North round to West, and across the map columns, row by row. There are five picsubs to each view (If you only need 4, repeat the last one twice!).

To exit this option before the landscape is complete press just ENTER on the "row?" prompt.

2. Save
This option will allow you to save the data array complete, or incomplete for reloading later. In the former case press "y", otherwise "n" and follow the normal SAVE prompts.

3. Load
First enter the width and height of your landscape map, then follow the LOADing prompts ("" loads in the first array on the tape). After the LOAD you will jump straight into the assembler option at the next view to compose.

4. View/Edit
This allows you to move around your landscape, and make minor alterations. The first view shown will be at grid 1,1 looking north. Pressing the cursors will give you the view in that

PROGRAM 1

```
   1 LET q$="": LET a=1: LET b=2
 : LET c=3: LET d=4: LET e=5: LET
 f=6: LET g=7: LET z=8: LET i=9:
 LET c$=CHR$ 22+CHR$ 7+CHR$ 0+"_"
 : FOR j=65351 TO 65535: INPUT p:
 POKE j,p: NEXT j
   9 CLS : INK 0: PRINT c$: FOR
j=1 TO (p-1)*6 STEP 6: LET v=VAL
 a$(j TO j+a): LET h=VAL a$(j+b
TO j+c): GO SUB VAL a$(j+d)*10+
VAL a$(j+e): NEXT j: RANDOMIZE U
SR 65351: INK 7: RETURN
  10 PRINT AT v,h+d;"NO";AT v+a,
h+b;"LKMP";AT v+b,h+a;"N  NOMNM"
;AT v+c,h;"N  N P ONM": RETURN
  11 PRINT AT v,h;"EEEEEEE";AT v
+a,h;"HFFFFFFG": RETURN
  12 PRINT AT v,h;"LUJLJ";AT v+a
,h+b;"K  I": RETURN
  13 PRINT AT v,h;"AC";AT v,h; O
VER 1;"__";AT v,h;".": RETURN
  14 PRINT AT v,h;"QT";AT v+a,h;
"ST";AT v+b,h;"RT";AT v+b,h; OVE
R a;"_": RETURN
  15 PRINT AT v,h;"LUJ";AT v+a,h
;"IUK": RETURN
  16 PRINT AT v,h;"______";AT v+a
,h;"TTTTTT";AT v+b,h;"TTTTTT": R
ETURN
  17 FOR l=NOT PI TO b: PRINT AT
 v+l,h;"▌": NEXT l: PRINT AT v+l
,h;"▟▙": RETURN
  18 PRINT AT v,h;"▄";AT v+a,h;
"▟▙": RETURN
  19 PRINT AT v,h+b;"S";AT v+a,h
+a;"▄█▄";AT v+b,h;"Q   O";AT v+c
,h;"R I P";AT v+d,h;"██████": RET
URN
  20 PRINT AT v,h+g;"NO";AT v+a,
h+e;"LKMP";AT v+b,h+d;"N  NOMNM"
;AT v+c,h+c;"N  N P ONO";AT v+d,
h+b;"Q LK  NMP P";AT v+e,h+b;"R
   Q   IJM";AT v+f,h+a;"NIJ   R
   NOIJ";AT v+g,h;"N    LK   LK
P KM": RETURN
  21 PRINT AT v,h;"ACACACACAC";A
T v+a,h;"TACACACACAC";AT v+b,h;"
ACACACACAC T";AT v+c,h;"TACACACA
CAC";AT v+d,h;"BT T T T T T";AT
v+e,h;" BDBDBDBDBD";AT v+b,h; OV
ER a;"I";AT v+c,h+i;" K": RETURN
  22 PRINT AT v,h+d;"_";AT v+a,h
+b;"LK IJLKIJ";AT v+b,h;"LK_   LK
LUJ IJ";AT v+c,h;"LK IJLK    IJ_"
;AT v+d,h+c;"LK": RETURN
  23 PRINT AT v,h+d;"_";AT v+a,h
+c;"A C";AT v+b,h+b;"N    M";AT v
+c,h+a;"A  K  O";AT v+d,h;"N
 MP";AT v+d,h; OVER a;"__AC_____"
: RETURN
  24 PRINT AT v,h+b;"_";AT v+a,h
+a;"NTO";AT v+b,h+a;"TTP";AT v+c
,h+a;"TT:T";AT v+d,h;"QQ  T";AT
v+e,h;"RSK T";AT v+f,h;"TS  O";A
T v+g,h;"TSO S";AT v+z,h;"TSP S"
;AT v+i,h; OVER a;"MSLK": RETURN
  25 PRINT AT v,h+a;"_LUUUJ__";A
T v+a,h;"N          M";AT v+b,h;"M
        _N";AT v+c,h+c;"IUUUK": R
ETURN
  26 PRINT AT v,h;"████████████
████": FOR l=a TO d: PRINT AT v+
l,h;"S▌█ █ █ █ █ █ █▐S": NEXT l:
 RETURN
  27 PRINT AT v,h+a;"▄";AT v+a,h
;"██": FOR l=b TO z: PRINT AT v
+l,h+a;"█": NEXT l: PRINT AT v+l
,h;"███": RETURN
  28 PRINT AT v,h+c;"S";AT v+a,h
+a;"▄████▄";AT v+b,h+a;"█████";AT
 v+c,h+a;"██████";AT v+d,h;"█████
█": RETURN
  29 PRINT AT v,h+z;"S";AT v+a,h
+z;"S";AT v+b,h+f;"▙█████▟";AT v+c
,h+c;"▙████████████▟";AT v+d,h+c;"█
██████████";AT v+e,h+b;"Q       |
    O";AT v+f,h+b;"R |          | P
";AT v+g,h+a;"Q         |      O";A
T v+z,h+a;"R |   ▙▄▄▄▟    | P";AT v
+i,h;"Q        ▌:S:▐       O";AT v+VA
L "10",h;"██████:S:██████": RE
TURN
9999>CLEAR 65350: LOAD "landcode
"CODE : REM - GO TO progstart
```

direction, whilst pressing "m" will move you that way (If there is no view there you will be told). Under each view is the string that produced it.

To make alterations to a view press "c". You'll get an information page plus the string in highlighted blocks. Remember the sequence in each block is row, column, then the picsub that you can identify from the details above. Use the left/right cursors to move the arrow to the first number you wish to alter, press SPACE, then move it to the last digit to alter and press SPACE again. Now enter the correct number sequence for replacement.

Designer landscape

Now to work on an example. Table C gives the data for the four views from grid 1,1 on my map. Enter these using option 1, then, using my map (or your own version), you can work out the design for the remaining grid positions/views (Surely you didn't expect me to do all the work!). Remember you can use 'n' in answer to the "?" at the end of a picsub positioning move to reposition it (or press 'a' to start the whole view over again), then use option 4 to wander through the whole scenario when it is complete, making corrections. Remember also when you make the final SAVE of the whole landscape to answer 'y' to the "landscaping complete?" prompt or you'll corrupt the last two entries!

What to do next? Ah well, the ball is in your court now to decide in what type of adventure you want to use your landscape. I can't do that for you but I can show you how to incorporate it into your programs and call up the landscapes.

You will need to CLEAR 65530 LOAD in the original program plus code that you SAVEd, then MERGE in your adventure program. Obviously your program must start after line 29 or you'll overwrite the utility (Also don't be tempted to alter any of the routine's line numbers or the GOSUB in line 9 won't operate correctly). Before you make any call to "the Landscaper" subroutine (using GOSUB 9) you must set r to the grid row of your map, k to the column and o to the orientation of the view you want (1 = N round clockwise to 4 = W). So to display the view looking west from the top centre grid square on our example map you would use the line

LET r = 1: LET k = 2 : LET o = 4: GOSUB 9

When you start the adventure you would need to decide where your adventurer was standing and which way he was facing, and set r, k and o to these values before GOSUB 9.

To allow him to move around your scenario you would need to include the following, probably as subroutines:—
To move in the direction faced:

1010 LET r = r + (o = 3) − (o = 1) : LET k = k + (0 = 2) − (o = 4) : GOSUB 9 : REM update row/column variables

To turn (cursor pressed in r$):

2010 LET 0 = (1 AND r$ = "7") + (2 AND r$ = "8") + (3 AND r$ = "6") + (4 AND r$ = "5"): GOSUB 9: REM relate value in o to keypress

An adventure isn't just wandering around a landscape so the values in r and k would also be used to check the predetermined positions of monsters/finds in arrays to see if you had bumped into a Balrog, or tripped over a Talisman or whatever.

Obviously you would SAVE the final adventure to LINE 9999 so as to LOAD in the code automatically (which would be SAVEd after it on the tape). You must also CLEAR 65350 before LOADing in the program to play. You could have the machine do this for you by LOADing first a short driver program

10 CLEAR 65350 : LOAD ""

which autostarts, lowers RAMTOP, then loads in the main adventure, which itself autostarts and LOADs n the code above the already lowered RAMTOP.

Bye for now.

PROGRAM 2

```
 100>LET y=0: PRINT '" ASSEMBLE/
EDIT"''" CONT to continue an ass
embly"
 101 IF r$="3" THEN LET y=1
 102 INPUT "width of landscape g
rid? "; LINE r$: GO TO (13 AND r
$="c")+(797 AND r$="")+103
 105 LET k=VAL r$
 106 INPUT "height "; LINE r$
 109 LET r=VAL r$
 110 DIM l$(k,r,4,30): LET m1=1:
 LET n1=m1: LET o1=n1: IF y=1 TH
EN LET y=0: RETURN
 115 FOR m=m1 TO k: FOR n=n1 TO
r: FOR o=o1 TO 4
 116 CLS : PRINT c$
 119 LET a$="": FOR p=1 TO 5
 120 INPUT "loc ";(m);",";(n);"
:view ";(o);" :pic ";(p);" row?
"; LINE v$: IF v$="" THEN GO TO
900
 124 IF LEN v$=1 THEN LET v$="0"
+v$
 130 INPUT "loc ";(m);",";(n);"
:view ";(o);" :pic ";(p);" col?
"; LINE h$
 134 IF LEN h$=1 THEN LET h$="0"
+h$
 140 INPUT "loc ";(m);",";(n);"
:view ";(o);" :pic ";(p);" pic?
"; LINE q$
 142 IF q$="h" THEN GO TO 190
 144 IF LEN q$=1 THEN LET q$="0"
+q$
 150 LET h=VAL h$: LET v=VAL v$
 154 GO SUB VAL q$
 155 PRINT #0;"pic ";(q$);" O.K.
 at ";(v);",";(h);"?(ENTER/n/a)"
: PAUSE 0
 156 IF PEEK 23556=78 AND p<>1 T
HEN GO SUB 9: GO TO 120
 157 IF PEEK 23556=78 AND p=1 OR
 PEEK 23556=65 THEN GO TO 116
 159 LET a$=a$+v$+h$+q$: NEXT p:
 LET l$(m,m,o)=a$: NEXT o: LET o
1=1: NEXT n: LET n1=1: NEXT m: G
O TO 900
 190 LET t=25: CLS : PRINT ,"Nea
r";TAB t;"Far"''"Mountain","10";
TAB t;"20"'"Forest","11";TAB t;"
21"'"Hills","12";TAB t;"22"'"Cav
ern","13";TAB t;"23"'"Monolith",
"14";TAB t;"24"'"Lake","15";TAB
t;"25"'"Henge","16";TAB t;"26"'"
Tower","17";TAB t;"27"'"Keep","1
8";TAB t;"28"'"Castle","19";TAB
t;"29": IF q$<>"h" THEN RETURN
 191 PRINT ''"Horizon at row 7":
 INPUT "Press ENTER to return",
LINE r$: GO SUB 9: GO TO 140
 200 PRINT '" SAVE DATA": PRINT
#0;"Landscaping complete? (y/n)"
: PAUSE 0: IF PEEK 23556=13 THEN
 GO TO 900
```

▶▶

```
 202 IF PEEK 23556<>89 THEN LET
l$(r,k,1)=STR$ m: LET l$(r,k,2)=
STR$ n: LET l$(r,k,3)=STR$ o
 205 INPUT "Title?"' LINE r$: SA
VE r$ DATA l$(): PRINT '"VERIFY"
  VERIFY r$ DATA l$(): GO TO 115

 300 PRINT '" LOAD DATA": GO SUB
 101
 305 INPUT "Title? ";r$: LOAD r$
 DATA l$(): LET m1=VAL l$(r,k,1)
: LET n1=VAL l$(r,k,2): LET o1=V
AL l$(r,k,3): GO TO 115
 400 GO TO 5000
 500 PRINT ''" Use GOTO 900 to r
estart": STOP
 900 CLS : PRINT '" OPTIONS"''''"
 1 Assemble/Edit"''" 2 Save Data
"''" 3 Load"''" 4 View"''" 5 Sto
p"''''" Press option number': PA
USE 0: LET r$=INKEY$: CLS : GO T
O VAL r$*100
5000 LET p=6: LET w=1: LET x=1:
LET y=1: GO SUB 5200
5001 PAUSE 0: LET r$=INKEY$: IF
PEEK 23556=13 THEN GO TO 900
5005 IF r$="c" THEN GO TO 5300
5010 IF r$="m" THEN GO TO 5100
5020 LET y=(1 AND r$="7")+(2 AND
 r$="8")+(3 AND r$="6")+(4 AND r
$="5")
5030 GO SUB 5200: GO TO 5001
5100 IF w=1 AND y=1 OR w=r AND y
=3 OR x=1 AND y=4 OR x=k AND y=2
 THEN GO TO 5001
5110 LET w=w+(y=3)-(y=1)
5120 LET x=x+(y=2)+-(y=4)
5130 GO SUB 5200
5140 GO TO 5001
5200 LET a$=l$(w,x,y): IF a$(1)=
" " THEN CLS : PRINT "loc ";w;",
";x;": view ";y;" Empty": RETURN
5210 GO SUB 9: PRINT #0;AT 0,0;"
loc ";w;",";x,"view ";y'a$: RETU
RN
5300 LET po=0: GO SUB 190: PRINT
 ''''" ";: PRINT a$(1 TO 6); INVE
RSE 1;a$(7 TO 12); INVERSE 0;a$(
13 TO 18); INVERSE 1;a$(19 TO 24
); INVERSE 0;a$(25 TO 30)'" ↑"''
"Use cursors to indicate start"'
"then press SPACE"
5301 IF INKEY$="" THEN GO TO 530
1
5302 LET po=po+(INKEY$="8" AND p
o<29)-(INKEY$="5" AND po>0)
5303 IF INKEY$=" " THEN LET p1=p
o+1: GO TO 5315
5304 IF INKEY$="m" THEN GO TO 90
0
5305 PRINT AT 16,po;" ↑ "
5306 GO TO 5301
5315 FOR l=1 TO 50: NEXT l
5320 PRINT AT 18,24;"end "
5321 IF INKEY$="" THEN GO TO 532
1
5322 LET po=po+(INKEY$="8" AND p
o<29)-(INKEY$="5" AND po>0)
5323 IF INKEY$=" " THEN LET n1=p
o+1: GO TO 5330
5324 PRINT AT 16,po;" ↑ "
5325 GO TO 5321
5330 IF n1<p1 THEN GO TO 5300
5370 INPUT "New string?" LINE r
$: LET l$(w,x,y,p1 TO n1)=r$: GO
 SUB 5200: GO TO 5001
8999 STOP
9000 LET q$="": LET a=1: LET b=2
: LET c=3: LET d=4: LET e=5: LET
 f=6: LET g=7: LET z=8: LET i=9:
 LET c$=CHR$ 22+CHR$ 7+CHR$ 0+"
: INK 7: PAPER 0: CLS : BORDER 5
: GO TO 900
9999 CLEAR 65350: LOAD "landcode
"CODE : GO TO 9000
```

Midi Patch Sequencer

A midi utility by Ray Elder for Spectrum music makers.

This program was born out of the need to change patches quickly from a single controlling unit in the studio. One of our Spectrum computers was used and the end result was modified to work on all the different MIDI interfaces we use.

The program printed will work with the MICON (XRI systems), EMR, SIEL, JMS, UPSTREAM or E+MM Spectrum Midi interfaces.

There are three individual modes of operation to this program and on loading you will be asked which method you want to use. From each mode there is the option to return to the first menu in order that you can use one of the others instead.

The modes are:

1. **Single** patch. A patch is sent to the instrument by typing in the number and pressing the ENTER key. The change is sent instantly and the information is not stored, enter one of the other keys as instructed to return to the menu.

   You can change the channel by entering 'c' at any input stage.

2. **Preset** patches. 10 preset patches can be stored and are sent instantly on selection of the respective number key. Initially all presets are set to 0 and you will have to enter your own choice by following the on screen instructions.

3. **Sequence** patches. Up to 10 patches may be entered when you first enter this section. These are sent one at a time every time a key is pressed, any key may be used except the P and R keys which have the special functions of allowing you to reset the sequence or return to the main menu.

   Once the end of a sequence is reached then the program loops back to the start of the sequence and begins again, eg. if two patches, say 20 and 39 were entered then by repeated pressing of a key they will be sent alternatively until the user decides to stop.

In all sections you can change the channel number whenever an input is requiring ENTER to be pressed is expected. Just type 'c' and press enter to access the channel options.

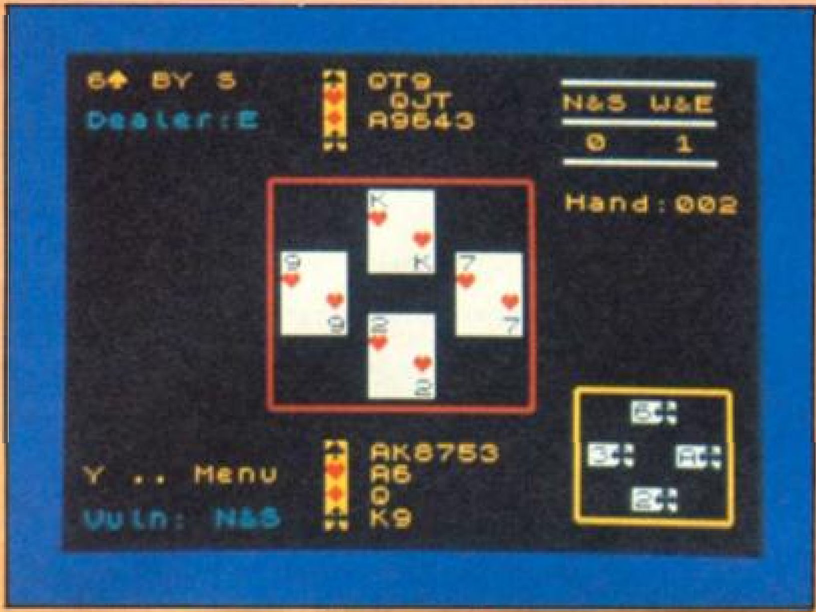
Listing 1

```
  20 CLS : PRINT AT 1,10;"PATCH
SEND.";AT 4,3;"1. MICON (XRI sys
tems)";AT 6,3;"2. EMR system";AT
 8,3;"3. SIEL, JMS";AT 10,3;"4.
UPSTREAM, E+MM"
  30 PRINT AT 14,6;"Press a key
1 to 4"
  40 LET g$=INKEY$: IF g$<"1" OR
 g$>"4" THEN  GO TO 40
  50 LET stat=63+(128 AND g$>"1"
)-(32 AND g$>"2"): LET trans=191
+(64 AND g$>"2")
  60 GO SUB 9000: GO TO 500
  99 STOP
 100 INPUT (m$); LINE p$
 110 IF p$="c" AND NOT flag THEN
  GO TO 200
 115 IF p$="r" AND NOT ret THEN
 LET ret=1: LET p=0: RETURN
 120 IF p$="" OR LEN p$>2 THEN
GO TO 100
 125 IF p$="e" THEN  LET end=1:
LET p=1: RETURN
 130 FOR i=1 TO LEN p$: IF p$(i)
<"0" OR p$(i)>"9" THEN  GO TO 10
0
 140 NEXT i
 150 IF VAL p$<0 OR VAL p$>max T
HEN  GO TO 100
 160 LET p=VAL p$: RETURN
 200 CLS : PRINT AT 4,8;"Change
channel number.";AT 6,9;"Present
 channel = ";chan
 210 LET flag=1: LET m$="Enter c
hannel number 1 TO 16 ": LET max
=16
 220 GO SUB inp: IF p<1 THEN  GO
 TO 220
 230 LET chan=p: LET ctrl=191+ch
an
 240 RETURN
 500 CLS : PRINT AT 1,8;"MIDI PA
TCH SEND";AT 6,6;"1. Single inpu
t";AT 8,6;"2. Preset patches";AT
 10,6;"3. Sequence of patches"
 510 PRINT AT 16,6;"Press a key
1,2 or 3"
 520 LET g$=INKEY$: IF g$<"1" OR
 g$>"3" THEN  GO TO 520
 530 GO SUB 1000*VAL g$
 540 GO TO 500
1000 CLS : PRINT AT 6,10;"Patch
change ";AT 8,10;"Last patch=";p
at;AT 10,10;"Channel No.=";chan;
AT 16,0;"Enter C to change chann
el numberor    R to return to th
e menu."
1010 LET ret=0: LET flag=0: LET
m$="ENTER PATCH NUMBER 1 to 99 "
: LET max=99
1015 IF ret THEN  RETURN
1020 GO SUB inp: IF p<0 OR flag
THEN  LET flag=0: GO TO 1000
1025 IF ret THEN  RETURN
1030 LET pat=p
1035 OUT trans,ctrl: OUT trans,p
1040 GO TO 1000
2000 CLS : PRINT AT 0,8;"PATCH C
HANGE PRESETS";AT 2,11;"CHANNEL
No.";chan;AT 5,0;"KEY No.   ";AT
9,0;"PATCH No."
2010 FOR i=1 TO 10: PRINT AT 6,i
*3-1;i-1;AT 8,i*3-1;p(i): NEXT i
2020 PRINT AT 12,0;"Press key";A
T 14,5;"0 TO 9 to send patch";AT
 16,5;"P to redefine patch prese
t";AT 18,5;"R to return to the m
enu"
2025 LET flag=0
2030 LET g$=INKEY$: IF g$<>"p" A
ND g$<>"r" AND (g$<"0" OR g$>"9"
) THEN  GO TO 2030
2040 IF g$="p" THEN  GO TO 2500
2050 IF g$="r" THEN  RETURN
2060 OUT trans,ctrl: OUT trans,p
(VAL g$+1)
2070 GO TO 2030
2500 LET m$="Enter preset number
 0 to 9 ": LET max=9
2510 GO SUB inp: IF flag THEN  G
O TO 2000
2520 LET x=p+1: LET m$="Enter va
lue of preset "+STR$ p+"
 (1 to 99) ": LET max=99
2530 GO SUB inp: IF p<1 THEN  GO
 TO 2530
2535 IF flag THEN  GO TO 2000
2540 LET p(x)=p: PRINT AT 8,x*3-
1;p
2550 GO TO 2030
3000 CLS : LET x=1: LET end=0: D
IM q(10): LET flag=0
3010 LET max=99: LET m$="Enter p
atch No."+STR$ x+" (e to End) "
3020 GO SUB inp: IF end  AND x=1
 THEN  LET end=0: GO TO 3020
3025 IF flag THEN  LET flag=0: C
LS : FOR i=1 TO x-1: PRINT AT i*
2,8;"PATCH ";i;"=";q(i): NEXT i:
 GO TO 3010
3030 IF NOT end THEN  PRINT AT x
*2,8;"PATCH ";x;"=";p: LET q(x)=
p: LET x=x+1: IF x<11 THEN  GO T
O 3010
3035 LET x=x-1
3040 CLS : PRINT AT 0,10;"PATCH
SEQUENCES": FOR i=1 TO x: PRINT
AT 4,i*3-1;i;AT 6,i*3-1;q(i): NE
XT i
3050 PRINT AT 8,0;"Press";AT 10,
6;"Any key for next patch";AT 12
,6;"R to return to menu";AT 14,6
;"P to set up a new sequence"
3060 FOR i=1 TO x
3065 PRINT AT 7,i*3-1; FLASH 1;"
^"; FLASH 0: LET y=i-1+(x AND i=
1): PRINT AT 7,y*3-1;" "
3067 IF INKEY$<>"" THEN  GO TO 3
067
3070 LET g$=INKEY$: IF g$="" THE
N  GO TO 3070
3080 IF g$="r" THEN  RETURN
3090 IF g$="p" THEN  GO TO 3000
3100 OUT trans,ctrl: OUT trans,q
(i)
3120 NEXT i
3130 GO TO 3060
9000 DIM p(10): LET chan=1: LET
ctrl=192: LET pnt=1: LET pat=1
9010 OUT stat,3: OUT stat,86: OU
T trans,176: OUT trans,124: OUT
trans,176: OUT trans,127
9020 LET ret=1: LET flag=0: LET
inp=100: RETURN
```

GAMES

COLOSSUS BRIDGE 4

CDS
£11.95



The game of bridge has received considerably less analysis by computer than might be supposed. After all, there are many chess programs available which are capable of beating the majority of players, so why isn't the same true for bridge?

There are a number of reasons but the main one is that as a game, it is incredibly difficult to analyse. Whereas a bishop can only move according to well defined rules, a bridge player has considerably fewer restrictions when it comes to making a bid or playing a card.

Side stepping some of the complexities of the game itself Colossus Bridge 4 is aimed fairly and squarely at the beginner.

Presentation wise, the program works well. Entering a bid or playing a card is simplicity itself and there are facilities to replay or rebid a hand if you want to study alternative lines. Deals can be listed to printer if you want to save a hard copy and you can input your own hands if desired. My only niggles here are that it is not too easy to distinguish between clubs and spades and the claim option which allows you to claim some or all of the remaining tricks without having to play the rest of the hand through. The problem here is that the program does not check the validity of your claim allowing you to cheat if you were unscrupulous enough so to do.

The standard of bidding is at best average. Allegedly following the Acol system — the one used by most British players, it copes well for the first round but then tends to get into murky waters. Wild leaps with inadequate trump support but a strong hand are commonplace but then it doesn't upgrade weaker hands with redeeming features such as good trumps and an unbalanced distribution. Once you get used to its style though, you should end up in the right contract on about 6-7 deals out of 10 where your side holds the balance of the high cards. The program's card play again isn't too hot, especially when defending and several of my contracts were allowed to be made when the computer simply failed to cash winners.

That's all well and good you say, but you know more or less what you are doing. How does the program rate for a beginner? The answer is not too bad at all. The tape is packaged with a book entitled Begin Bridge by Geoff Fox, one of the foremost bridge teachers and the combination of the two will prove an admirable introduction for anyone new to the game.

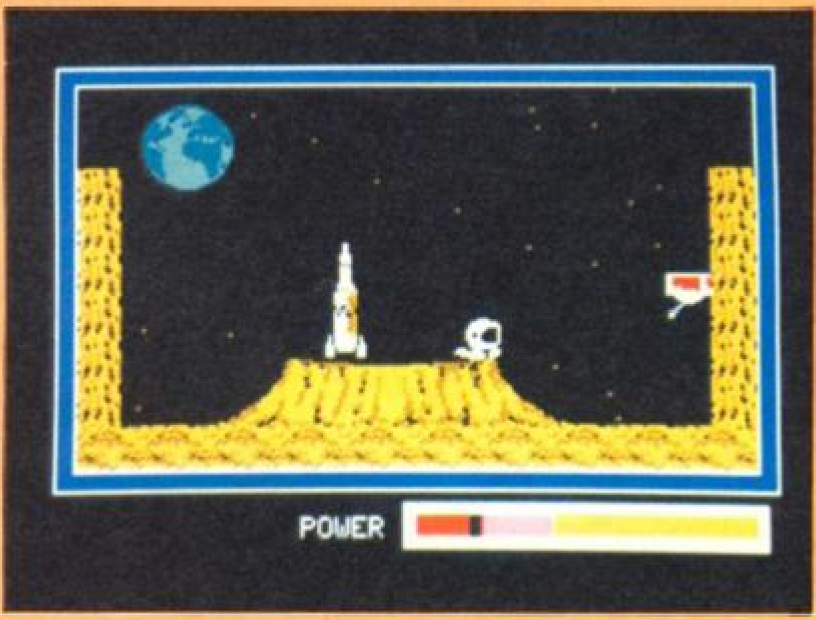
Read up on the basics of the game first and then get a feel for the mechanics of the game via the program. One last grumble though. Side two of the tape contains ten demonstration hands. Apart from the fact that the program is badly bugged so you cannot actually play the hands through, the choice of deals is also poor. Elimination and endplays together with advanced avoidance techniques have no place in a beginner's package.

To conclude then, a reasonable introduction for the beginner who finds it impossible to learn from books alone but the more experienced player is likely to get extremely annoyed at lack of a real challenge.



PHANTOMAS

Codemasters
£2.99



Phantomas is a mutant. No ordinary mutant mind, but one specialising in robbery, plunder and pillage. Intended for use in the battle of the red moons on Alpha Centauri, his talent for eluding capture has worked against his manufacturers and Phantomas escaped, becoming a free agent.

Stories of great wealth abounded in the local press and one that particularly caught his eye was details of the miser Goldter's horde secreted somewhere in his mansion on the clone planet Earth-Gamma. Naturally, the mansion was designed to be burglar proof, but a little thing like that had never stopped Phantomas before.

The mansion comes in three different parts. Parked outside are a variety of flying craft which will transport you to assorted outposts. Then there is the palace itself and an underground complex. In order to 'liberate' the strongbox, Phantomas must find and throw thirty-six levers scattered about the house and grounds. There are also jewels to be found but you must first solve two riddles if you are to succeed. The game features many other little twists. When you fly off in the helicopter, you see a switch lying next to a pile of machinery. Pull the switch and you are immediately pursued by a giant snowball. The only way of escape is by rapidly hitting two keys a la Daley Thompson's Decathlon.

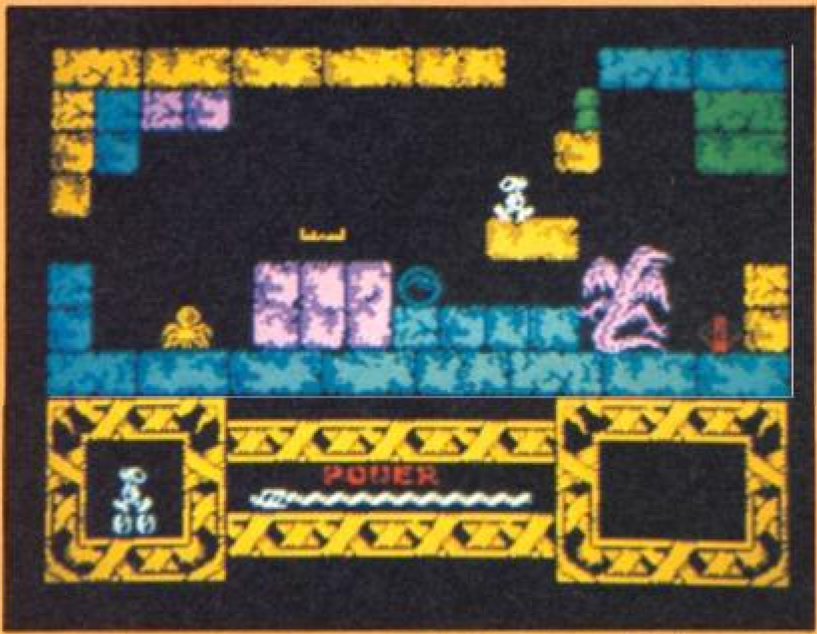
Energy cubes lie all over the place to replace whatever you lost in unfortunate collisions with the nasties. Another nice touch which separates this from run of the mill platform games is that Phantomas has two distinct types of jump available to him — a long jump for added distance and a high jump that enables him to leap tall buildings (well small obstacles really) at a single leap.

There seems to be a trend at the moment for mixing platform games with arcade adventures in order to produce a new hybrid and Phantomas is definitely towards the top of the range. Full of original and inventive ideas, the eighty screens will keep you hunched over your keyboard for many a long hour.



VAMPIRE

Codemasters
£2.99



Bored with his daily existance of drugs and therapy sessions, Brok the Brave is more than pleased to hear from an emissary from Sol 1 who proposes an interesting mission.

Earth and its space stations need liberating from the terrors of the evil Count Dracula who has the entire population in a state of continual nightmare. Glad of any chance to escape from his humdrum life on the high security planet Hawkland, Brok readily accepts the challenge and is transported to the entrance of the Count's castle.

Vampire is a platform game with arcade adventure overtones set over some eighty screens. Your first problem is one of survival. Starting with only one life, contact with various nasties or flying arrows soon reduces your energy. Mistimed leaps can plummet you into pits from which there is no escape and areas of the castle seem inaccessible until you collect certain items necessary to move false walls. Flashing items of food provide welcome extra lives and there are keys to be collected en route as well.

Just finding Dracula is not enough. He must be destroyed using the traditional stake, hammer and cross but you must also solve a complex riddle included in the instructions which involves finding a room where no light will reach.

As Brok moves around the castle (there are both surfaces and underground areas to explore), he can leap obstacles in both large and small bounds which brings another element into the gameplay. Jumping is done automatically in the direction that you are facing so that there is no need to find those tricky diagonals on your joystick. There is some latitude for moving Brok whilst in mid-leap.

The game's graphics are excellent featuring highly coloured backdrops and a whole variety of castle furniture. Playing Vampire is difficult — your route is not intended to be an easy one — and I found considerable initial frustration at my inability to get anywhere before I lost all my energy. But perseverance pays off and slowly, further areas of the castle are revealed. A good, challenging game.





HERCULES

Alpha Omega
£1.99

Occasionally, a game comes along that proves the point that you don't have to have great graphics for it to be addictive. I originally saw Hercules on the C64 and I thought that no-one could come up with a game that had worse graphics. Well the people who converted it to the Spectrum have managed!

So what is it about Hercules that makes it addictive? Well, it is a platform game but with the added interest of problems to be solved as well as monsters to be leapt etc.

You play the part of Hercules as you try to complete his twelve labours. Each labour is spread over several screens and the first eleven appear in a random order. Only if you succeed in completing all of them will you have the chance to have a crack at the twelfth.

Your first problem on each screen is usually one of time. The platform you start on has the annoying habit of bursting into flames and frying you to a crisp if you stand around waiting for more than a couple of seconds. This forces you into action which is usually equally lethal. Ropes and platforms collapse under you plummeting you to a fiery grave.

Frequently, the route to your objective — a large door — isn't obvious at all and you have to take a wild plunge into the unknown hoping that a platform will miraculously appear under you. You soon get used to ignoring the platforms originally on the screen as they tend to be red herrings. This leads a lot of people to claim that the game is too random but it only needs for you to complete a couple of the fifty plus screens to get the hang of what's going on and you are hooked.

Presentation wise, the game is dreadful. The chosen keys are unplayable with no redefine option although you can use a joystick. When it says 'Press P to start', it actually means Caps shift and P and as already mentioned, the graphics are dreadful — the crudity of the C64 version with added garishness. But, for all that, I do keep going back and playing it again and again.





ICE TEMPLE

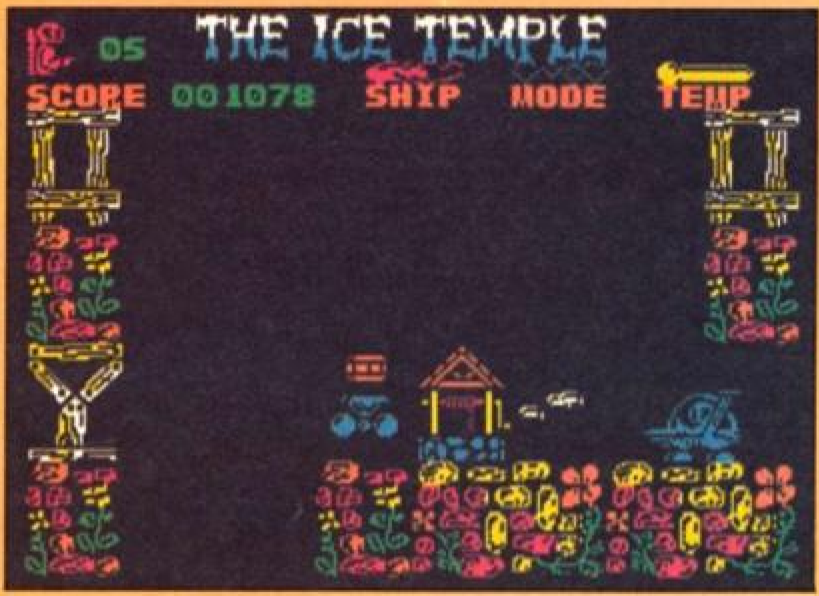
Bubble Bus
£8.95

If you've played Bubble Bus' earlier game, Starquake, you're likely to get a touch of the deja vus when you load up Ice Temple, as their latest game is an arcade adventure in a very similar mould.

You control one of those galactic types who dons his spacesuit and jetpack and sets off to explore the caves beneath the surface of some planet or another, searching for useful objects and blasting lots of aliens along the way (stop me if you've heard this one before).

This time around you play a character called Nick Razor, who is searching for his speedy Space Cruiser which has been stolen by aliens and hidden away in the depths of their Ice Temple. However, finding the cruiser isn't your sole task for there is a thermionic reactor at the heart of the temple which the aliens are using as a power source while preparing to attack the Earth. So, you leap into your cruiser and start zooming around the caves in search of the reactor.

The temple is pretty extensive, and at certain points along the way there are teleport beams which make a nice 'bwoooiiing sound as they send you off into some far corner of the temple. As usual in this type of game, the moment you enter a cave all manner of bouncing sprites appear out of thin air and if they come into contact with you they'll drain the heat from your space suit until you freeze to death, losing one of your five lives. As you'd expect there are various objects in the caves which will help you on your way — extra fuel can-



nisters, gems which make you temporarily invulnerable, magnetic cards, smart bombs and the like.

The graphics are colourful and well animated, but there's nothing about the game that makes it stand out from all the others of this type. Bubble Bus have produced good games in the past, but their recent releases just don't seem to have progressed from where they were about a year ago. As a result, Ice Temple is a competent game, but it just seems a bit old hat these days.



FROST BYTE

Mikrogen
£8.95

Mikrogen's latest is a little disappointing I'm afraid. At first it looks like a colourful arcade adventure in the style of Nodes of Yesod, Starquake and so on, in which you have to trek through underground caverns searching for various objects.

Graphically it's nice enough, bright colours and smooth animation, but after playing the game for a short while you realise that it's one of those games that allows you very little margin for error and can be enormously irritating.

There's an element of novelty in the fact that you control a worm-like creature called a Kreezer, and instead of the usual left/right and flying movements that are standard in these sort of games, the Kreezer moves along with a sideways end-over-end movement... He can also jump, but this action is limited to jumping straight up with only a very limited ability to move left or right as he comes back down again. This means that getting past most of the traps and monsters requires enormous accuracy in positioning and timing and at times the whole thing becomes an annoyingly frustrating matter of trial and error as you attempt to work out the correct manoeuvre.

The Kreezer's task is to rescue five other Kreezers who are locked up in different levels of the caves. But whereas in other games of this type the caves are often interconnected in a complex pattern which allows many different routes through them, in Frost Byte the caves of each level seem to be connected in a fixed sequence so that in order to master the later obstacles you have to go through the earlier stages over and over again, and this soon becomes a bit of a chore.

It's a shame, because there's an enjoyable game in here, but it's been spoiled by not being very well thought out.



FAT WORM BLOWS A SPARKY

Durell
£9.95

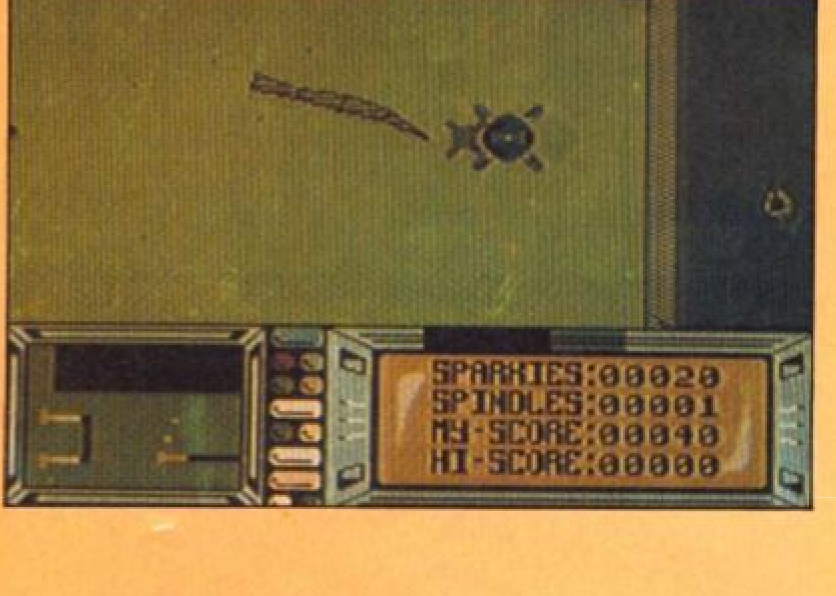
Oh dear. I've generally liked Durell's games in the past, and their recent Thanatos is one of my current favourites but Fat Worm has turned out to be about as enticing as the name suggests.

You play the fat worm of the title, a sluggish, spindly thing that is wandering around the circuit board of a Spectrum (cue for lots of puns about 'bugs' in the instructions). Along the way you have to climb up and down the ramps of the different databases to eat the floating triangles, and you have to eat fifty of these before finally locating the disc drive and cloning yourself. The triangles replenish your supply of 'blaster sparkies' which are used to kill the crawler bugs, and you also have a supply of 'burper sparkies' for killing the creeper bugs who are flying around in sputniks. If you think that it all sounds pretty daft you should try playing it...

The most interesting part of the game is the way the worm moves — he/she/it moves via alternate clockwise/anticlockwise swaying motions which are hard to control but actually create a realistic slug-like movement (mind you the swaying of the screen display made me feel sea sick).

The circuit board of the Spectrum is represented as a 3D overhead view, and from a programming and graphics point of view it's all quite clever. But from a fun and games point of view it all seems like a bit of a lost cause, and for £9.95 Durell have fallen short of their usual entertainment value.







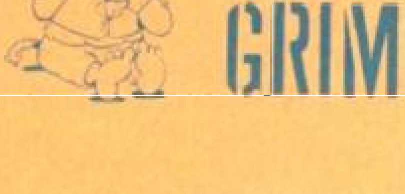
OBLIVION

Alpha-Omega
£1.99

This isn't "the best pure arcade game I've ever seen," as the cassette inlay claims, in fact it isn't anywhere near it. But if you're looking for a very simple, old-fashioned shoot 'em up of the sort that was popular about three years ago then this is probably right up your street.

Oblivion reminds me of a Jeff Minter Intergalactic Llamas type shoot 'em up. You control what looks like an emu who walks from left to right across screenfulls of ghosts and spaceships and aliens who are all approaching from the opposite direction and are ready to blast the tail feathers off you if you're not equally quick on the trigger. As I mentioned, it's all very Minter-ish, with a moving background of stars to give an outer space effect, and your emu bounces around quite nicely as he spits fire across the screen. It would be nice if you could fire and walk from left to right — you can slow down your walking, which results in you sliding backwards to the left hand side of the screen, but you don't actually change direction so you're a bit limited in manoeuvreability.

It's the sort of game that doesn't require amazing graphics to be effective — it's simply a question of whether or not you enjoy this type of simple target shooting. The game only supports a fixed set of keyboard controls or a Kempston joystick so you should bear that in mind if you're going to buy it.



DERESOLUTION

Chris Strangroom, who converted Palace Software's Antiriad to the Spectrum presents a sprite deresolution routine to incorporate into your programs.

The following program prints up the first five letters in the alphabet — as a sprite — and then deresolves.

If you alter the values of the labels between lines 11 and 16, you can change the position, width and depth of the sprite and the start position of its data. If you change the label RESOLUTION_TYPE, you can resolve the sprite onto the screen.

The program can be typed into most commercially available assemblers — some of the label names may need abbreviating — or if you have no assembler, the column of hex — starting line 23 — should be typed into a hex-loader. Better still, just incorporate the routine DERES into your own program.

```
1  0000                      HEADING ZX COMPUTING DERESOLUTION DEMONSTRATION PROGRAMME
2                            PL 59
3                            TOP 0
4                            MACLIST OFF
5                            FILLCHAR 0
6
7
8        EA60      PROG             EQU 60000      ;ASSEMBLY ADDRESS
9        22AA      PIXEL_ADD        EQU 22AAH      ;ROM ROUTINE TO CALCULATE SCREEN ADDRESS
10
11       3E08      SPRITE_GRAPHICS EQU 3E08H       ;WHEREVER YOUR SPRITE DATA IS
12       0008      RESOLUTION_TYPE EQU 8           ;RESOLVE = 8      DERESOLVE = 129
13       0001      WIDTH            EQU 1
14       0028      DEPTH            EQU 40
15       0078      X                EQU 120
16       0078      Y                EQU 120
17
18       0028      SPRITE_LENGTH   EQU WIDTH*DEPTH
19
20 0000
21 EA60                      ORG      PROG
22
23 EA60  3E 03     ZXCOMP: LD A,3
24 EA62  D3 5D             OUT (93),A
25 EA64  D3 7F             OUT (127),A
26 EA66  CD 89 EA          CALL SETUP
27 EA69  CD 6D EA          CALL MOVE
28 EA6C  C9                RET
29
30 EA6D  76        MOVE:   HALT
31 EA6E  CD CD EA          CALL XORSPR             ;UNPLOT SPRITE
32 EA71  CD B5 EA          CALL EXPAND             ;NORMALLY THIS WOULD BE DONE BEFORE UNPLOTTING - TWO BUFFERS WOULD B
                   E USED
33 EA74  DD 7E 04          LD A,(IX+4)
34 EA77  A7                AND A
35 EA78  C4 05 EB          CALL NZ,DERES


36 EA7B  CD CD EA          CALL XORSPR             ;PLOT UP SPRITE
37 EA7E  76                HALT
```

```
EA7F  76                   HALT                        ;HALTS ARE JUST A DELAY
EA80  3E 7F                LD A,7FH
EA82  DB FE                IN A,(0FEH)
EA84  1F                   RRA
EA85  DA 6D EA             JP C,MOVE
EA88  C9                   RET                         ;RETURN IF BREAK IS PRESSED

EA89  DD 21 7F EB  SETUP:  LD IX,SPRITE_TABLE          ;INITIALISE SPRITE AND PLOT IT
EA8D  DD 36 00 78          LD (IX),X
EA91  DD 36 01 78          LD (IX+1),Y
EA95  DD 36 02 01          LD (IX+2),WIDTH
EA99  DD 36 03 28          LD (IX+3),DEPTH
EA9D  DD 36 04 08          LD (IX+4),RESOLUTION_TYPE        ;THIS CAN BE EITHER RESOLVE (08H) OR DERESOLVE (01H)
EAA1  CD B5 EA             CALL EXPAND
EAA4  DD 7E 04             LD A,(IX+4)
EAA7  FE 08                CP 8
EAA9  CC 05 EB             CALL Z,DERES
EAAC  CD CD EA             CALL XORSPR
EAAF  06 14                LD B,20
EAB1  76           SETUP1: HALT
EAB2  10 FD                DJNZ SETUP1
EAB4  C9                   RET

EAB5  21 00 3E     EXPAND: LD HL,SPRITE_GRAPHICS       ;EXPANSION IS USUALLY A PRELUDE TO SHIFTING
EAB8  11 84 EB             LD DE,BUFFER
EABB  3E 00                LD A,0
EABD  DD 46 03             LD B,(IX+3)
EAC0  DD 4E 02             LD C,(IX+2)
EAC3  C5           EXPAN1: PUSH BC
EAC4  47                   LD B,A
EAC5  ED B0                LDIR
EAC7  12                   LD (DE),A
EAC8  13                   INC DE
EAC9  C1                   POP BC
EACA  10 F7                DJNZ EXPAN1
EACC  C9                   RET

EACD  DD 4E 00     XORSPR: LD C,(IX)                   ;SET UP PARAMETERS AND PLOT/UNPLOT SPRITE
EAD0  DD 46 01             LD B,(IX+1)
EAD3  CD AA 22             CALL PIXEL_ADD              ;HL=SCREEN ADDRESS
EAD6  DD 46 02             LD B,(IX+2)                 ;B=WIDTH
EAD9  04                   INC B
EADA  DD 4E 03             LD C,(IX+3)                 ;C=DEPTH
EADD  11 84 EB             LD DE,BUFFER
EAE0  CD E4 EA             CALL PLOT
EAE3  C9                   RET

EAE4  C5           PLOT:   PUSH BC                     ;A SIMPLE XOR SPRITE ROUTINE
EAE5  4D                   LD C,L
EAE6  1A           PLOT1:  LD A,(DE)
EAE7  AE                   XOR (HL)
EAE8  77                   LD (HL),A
EAE9  13                   INC DE
EAEA  2C                   INC L
EAEB  10 F9                DJNZ PLOT1
EAED  69                   LD L,C
EAEE  C1                   POP BC
EAEF  0D                   DEC C
EAF0  C8                   RET Z
EAF1  24                   INC H
EAF2  7C                   LD A,H
EAF3  E6 07                AND 7
EAF5  C2 E4 EA             JP NZ,PLOT
EAF8  7D                   LD A,L
```

```
EAF9   C6 20                    ADD A,32
EAFB   6F                       LD L,A
EAFC   38 E6                    JR C,PLOT
EAFE   7C                       LD A,H
EAFF   D6 08                    SUB 8
EB01   67                       LD H,A
EB02   C3 E4 EA                 JP PLOT

EB05   DD 7E 04         DERES:  LD A,(IX+4)
EB08   FE 89                    CP 137
EB0A   20 04                    JR NZ,DERES1
EB0C   DD 36 04 88              LD (IX+4),136   ;NORMALLY THE SPRITE WOULD BE KILLED AT THIS POINT
EB10   21 35 EB         DERES1: LD HL,DMASKS-8
EB13   87                       ADD A,A
EB14   87                       ADD A,A
EB15   87                       ADD A,A
EB16   5F                       LD E,A
EB17   16 00                    LD D,0
EB19   19                       ADD HL,DE       ;HL=START OF MASK FRAME
EB1A   11 84 EB                 LD DE,BUFFER    ;DE=SPRITE BUFFER START
EB1D   0E 0C                    LD C,(SPRITE_LENGTH+SPRITE_LENGTH/WIDTH)/7+1
EB1F   06 07                    LD B,7
EB21   C5               DERES2: PUSH BC
EB22   E5                       PUSH HL
EB23   1A               DERES3: LD A,(DE)
EB24   A6                       AND (HL)
EB25   12                       LD (DE),A
EB26   23                       INC HL
EB27   13                       INC DE
EB28   10 F9                    DJNZ DERES3
EB2A   E1                       POP HL
EB2B   C1                       POP BC
EB2C   0D                       DEC C
EB2D   20 F2                    JR NZ,DERES2
EB2F   DD CB 04 7E              BIT 7,(IX+4)
EB33   28 04                    JR Z,DERES4
EB35   DD 34 04                 INC (IX+4)      ;IF BIT 7 IS SET, INCREASE MASK (DERES)
EB38   C9                       RET
EB39
EB39   DD 35 04         DERES4: DEC (IX+4)      ;IF BIT 7 IS CLEAR, DECREASE MASK NUMBER
EB3C   C9                       RET


EB3D   F7 7F BF DF FB   DMASKS: DB 11110111B,01111111B,10111111B,11011111B,11111011B,11111110B,11111101B,0     ;1
EB42   FE FD 00
EB45   77 DD 5F F5 EB           DB 01110111B,11011101B,01011111B,11110101B,11101011B,11101110B,11111010B,0     ;2
EB4A   EE FA 00
EB4D   73 DC 1F E5 E3           DB 01110011B,11011100B,00011111B,11100101B,11100011B,11101100B,01111010B,0     ;3
EB52   EC 7A 00
EB55   63 CC 5A A5 55           DB 01100011B,11001100B,01011010B,10100101B,01010101B,01010101B,10101010B,0     ;4
EB5A   55 AA 00
EB5D   62 CB 1A B5 6C           DB 01100010B,11001000B,00011010B,10000101B,01101100B,10001010B,11001001B,0     ;5
EB62   8A C9 00
EB65   90 09 90 C0 22           DB 10010000B,00001001B,10010000B,11000000B,00100010B,00010010B,00100100B,0     ;6
EB6A   12 24 00
EB6D   08 80 02 10 40           DB 00001000B,10000000B,00000010B,00010000B,01000000B,00000100B,00000001B,0     ;7
EB72   04 01 00
EB75   00 00 00 00 00           DB 0,0,0,0,0,0,0,0                                                             ;8
EB7A   00 00 00

EB7D   0000             STACK:          DW      0
EB7F                    SPRITE_TABLE:   DS      5
EB84                    BUFFER:         DS      SPRITE_LENGTH+SPRITE_LENGTH/WIDTH

EBD4                            END
```

FIRELORD

Firelord
Hewson
£8.95

Hewson's medieval magic arcade adventure has our reviewer under its spell.



Firelord has all the elements we have to expect from Steve Crow, author of Starquake and Wizards Lair, but it's also a dazzling progression from his previous work.

The plot is fairly standard — you play Sir Galaheart and must collect the charms and free the land from the Evil Queen who has put a fiery curse on the land with the power of the magical Firestone. Don't be put off by a familiar plot as Firelord is executed faultlessly and will provide days of entertainment on many different levels.

There are a multitude of things to do in this 512 screen feudal word but you won't get very far unless you first locate an enchanted crystal which will give you some firepower against the bands of knights who roam the land. Objects are scattered throughout the landscape and these are necessary to give you bartering clout to obtain the spells you need. So it's not just a matter of "collect the right objects and you are home and dry" and the barter sequences are a very clever addition. Enter a house and you are transferred to an icon driven screen which shows you the house owner (witch, peasant, etc) by way of an animated inset picture of a face, what objects you have and what objects are on offer.

You could of course go for a straight transaction but there is also an icon which gives you the chance to get away with what you are after for nothing. You have to be pretty fast with the cursor to do this and if you mistime it retribution is swift.

If you get caught red-handed you are brought before the judge and may forfeit up to three of your five lives if you fail to come through a sub game that requires even greater split second timing than carrying off the crime itself which is a nice touch of poetic justice.

Even after long periods of play I found I'd barely scratched the surface of the game but had been enthralled throughout and that to me is a sure sigh that Firelord will be returned to again and again.

The game features very attractive graphics, smooth animated figures, and a complexity of gameplay that is pitched at just the right frustration level. It holds more than enough surprises to captivate even the most accomplished arcade adventurer.



THANATOS



Thanatos
Durell
£9.95

Enter the dragon! Wizardry, chivalry and mythical lizardry as Thanatos takes to the sky.

Durell's latest game adds a twist to the old 'people versus big bad dragon' scenario, as this time around you play the dragon and, instead of threatening the distressed maiden you enlist her aid in searching for a cauldron and book of magic spells.

In some ways Thanatos is about as close to a graphical representation of Dungeons and Dragons as any game since Knight Lore, as the way in which you control your dragon figure helps to create more of the 'feel' of D&D than many of the more complex simulations.

At the start of the game you find yourself in control of the dragon, Thanatos, a large and excellently animated figure who can walk or fly over the countryside as he searches for the enchantress Eros, and the other items he needs to complete his quest. The



landscape is represented in a sort of 'layered' 3D perspective which lends a feeling of depth to the graphics, foreground objects being placed lower down the screen than background objects — a simple technique, but quite effective.

Initially you fly west (left to right across the screen) through caves where you have to avoid rockfalls, over the heads of foot soldiers who will attack you with bows and rocks, and over the sea to reach the first castle where Eros is waiting for you. Once past the first stage, there are three zones (caves, countryside and sea), but the caves become full of deadly spiders, the countryside guarded by ever more soldiers and the air above both land and sea is filled with other monsters, none of whom realise that a dragon's just gotta do what a dragon's gotta do.

All these attackers put a bit of a strain on the poor dragon's heart, which is depicted in the lower left hand corner of the screen. As the strain increases his heart beat speeds up until — if you don't slow down and rest for a bit — he drops dead of a heart attack. The hard part is finding somewhere quiet enough to rest as you won't be able to just sit down and ask the sea serpent not to attack you for a while.

The fun bit lies in the way you get to strike back. Like any dragon worth his or her salt, Thanatos can breath flame and frazzle those insignificant people on the ground if they get in his way. Also, as you get the hang of flying you'll find that you are able to pick up rocks and other objects and drop them in order to demolish targets on the ground. Thanatos doesn't have the sheer pace of some arcade games but, when you swoop down out of the sky, grab a soldier in your claws then fly to the top of the screen before carefully dropping him onto a group of other soldiers, it's hard to beat for sheer vindictiveness. I came unstuck here quite a few times because, sadist that I am, I sometimes spent so much time depopulating the countryside that I ran Thanatos into the ground before I'd got anywhere near completing the rest of the quest.

Burned out

When you're likely to have trouble is when you run out of flames, because you need a lot of firepower to burn down the castle doors in each stage of the game. There's a 'flame-ometer' in the bottom right of the screen, and when you run out you have no choice but to go back the way you came and try and find the witch. If you can eat her your flame will be recharged but she's guarded by a knight on a white horse who tends to come charging across points above the landscape where you can build up a bit of speed and just leave the dragon on autopilot for a while until he reaches the next stage. And on several occasions I've spent forever looking for the witch in an attempt to recharge the dragon's flame and I just haven't been able to find her. Maybe she's there somewhere, but it's frustrating not being able to go onto the next part of the game just because you can't find her.

And of course, wherever you have a Spectrum game with large multicoloured moving figures you're going to get those good old 'attribute clashes'. Admittedly, you can't hold Durell responsible for the machine's attribute problems, but they are responsible for the price, which at £9.99 is a bit pricey. But despite these points Thanatos is still highly addictive and well worth a Monster Hit.

the screen with a long pointed lance aimed at you.

The quality of the animation does a lot to enhance your enjoyment of the game, because with just a little bit of imagination you can easily imagine yourself flying over the heads of the people below, prior to trampling them into the ground. Mind you, the game isn't perfect as there are certain

MONSTER HIT

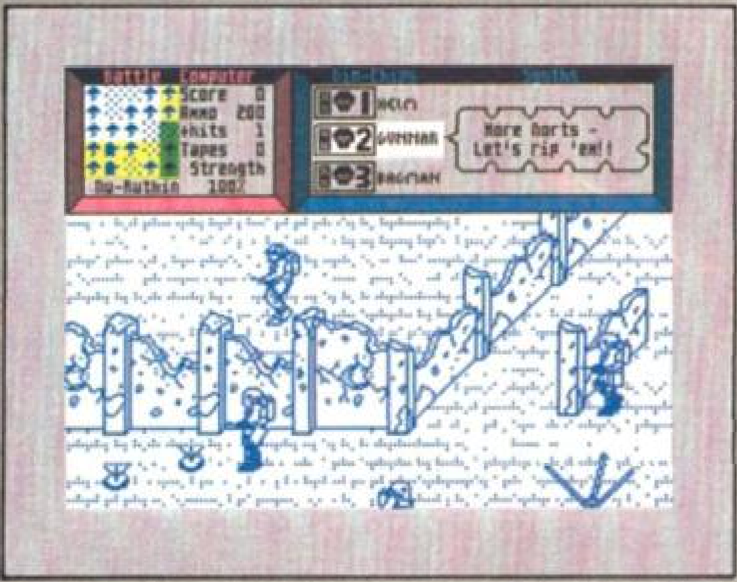


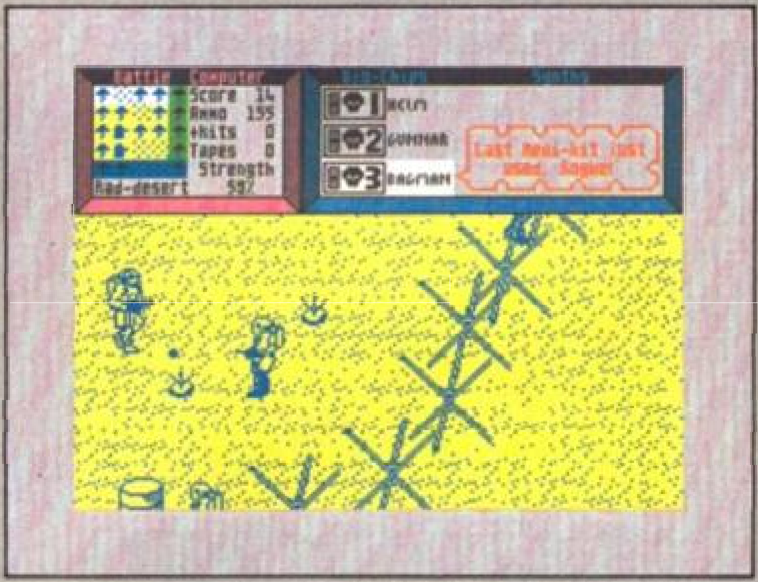


ROGUE TROOPER

Rogue Trooper
Pirahna
£7.95

2000AD comic hero, Rogue Trooper, blasts into action.





You are Rogue Trooper a genetically engineered infantryman created to withstand the piosonous chem-clouds of Nu Earth. You are also the sole survivor of the Quartz massacre when your regiment was betrayed and wiped out by the Norts.

Now your only thought is to find the traitor reponsible for the slaughter and revive our three buddies that are now mere bio-chips mounted on your gun, helmet and backpack.

These three chip buddies advise you during your misson as well as stir you on to more herioc deeds. Gunnar helps your firing but has a distinctly psychotic side to his nature and will constantly urge you to "blast some more Nort scum!" Helm gives you the odd hint and points you in general direction of medi kits and ammo stores and gets very excited when you find one of the eight vid tapes that will prove the identity of the traitor. Finally Bagman reports on current levels of ammo, kits and tapes and applies the medi kits if you're getting weak.

The war torn planet of Nu Earth is represented by 3D graphics illustrating the ruined cities, radiation deserts, graveyards, fuel dumps, nu forests and glass zones that are now swarming with Nort troops.

Rogue Trooper is controlled by joystick or redefinable keys and can explore the "real" landscape. This is obviously a development of Ultimate's original Knight Lore stystem tailor made to the ruins, trees, graves and barbed wire of Nu Earth.

Charging around with guns blazing will only have short lived success as you will be quickly cut down by the Nort troopers, auto-firing pillboxes and mines that have a nasty habit of exploding if you get too close to them.

The way to success and a longer life is to use the scenery as cover, only rushing into the open when you can get first shot. Gunner might not like these tactics but you don't have to take all of the avice the chip buddies hurl at you.

My best so far is seven of the eight vid-tapes so the shuttle will have to wait a little longer before taking me to victory.

The Norts may hve finally gunned me down but I took plenty of them with me. One thing's certain, I'll be back for more.



Ray Elder presents a program for efficient text storage.

TEXT MISER

Although the program as printed is nothing special, the end product will save a minimum of 15 bytes per message, on a 100 location adventure this becomes 1500+ bytes!

In simple terms the program takes a text and/or graphics input and stores it sequentially as bytes. The short 36 byte machine code routine then prints out the text as required.

The routine and code is completely relocatable and options to view the text before storage, re-entering it, and viewing all the stored text are included.

The main disadvantage is that editing of stored text is not possible (at the moment) so it is wise to plan the entries before using the program. Once you have entered the program, the machine code is built in — make sure the DATA line is accurate, just run it and follow the prompts. If you just press ENTER for the x,y co-ordinates and the INK and PAPER colours, the program defaults to PRINT AT 0,0;INK 0;PAPER 7.

At any time you can leave the text entering routine and view what you have stored so far, going back from the option page if you required to add more.

Details of the number of entries made and the length of the file are also given so you can keep an eye on how much space you have left. The program starts at 40000 but this can be lowered if you wish. From the option menu you can save the code to tape, it saves as "text" CODE 4000, length.

Using the text

The stored text can be used from either machine code programs or from basic, first it should be loaded to the required address, for example to load it to 50000 first CLEAR address-2 (49998) then LOAD ""CODE 50000, obviously the address+length (as given on the option page) must not exceed 65535. The reason for clearing two below the address is that the program uses the address-1 location to get the text item number.

For **BASIC** the most economical way is to set up a variable for the call address, eg. LET z=50000, and to print the required message POKE the item number into z-1. So to print the fifth message you would use POKE z-1,5: RANDOMIZE USR z.

As you may have gathered it is very useful to keep a list of your messages, menu option 1 is useful for this.

From **machine code,** stack the values of AF, BC, DE and HL if you want to preserve them, load BC with the routine's address — LD BC, 50000 — POKE the message number to the address-1 and CALL the address.

Advantages

Apart from being an economical way of storing text, it has the advantage that the text cannot be read by listing the basic program. The technique of setting up such a text "table" is one which has been used extensively before the advent of good old inefficient basic, where DIMming a string often means most of the string is wasted spaces.

Each message takes up only the length of that message plus eight bytes, the format of the stored message is:

1 byte, total length of entry
1 byte, paper token character
1 byte, paper colour
1 byte, ink token character
1 byte, ink colour
1 byte, AT token character
1 byte, Y co-ordinate
1 byte, X co-ordinate
n bytes, the character of the message

Listing

```
  10 CLEAR 39998
  20 FOR i=40000 TO 40035: READ
a: POKE i,a: NEXT i
  30 LET addr=40050
  40 LET no=0
 100 INPUT "Enter Y pos. 0-21 ":
 LINE y$: IF y$="" THEN  LET y$=
"0"
 110 IF VAL y$<0 OR VAL y$>21 TH
EN  GO TO 100
 115 IF INKEY$<>"" THEN  GO TO 1
15
 120 INPUT "Enter X pos. 0-31 ":
 LINE x$: IF x$="" THEN  LET x$=
"0"
 130 IF VAL x$<0 OR VAL x$>31 TH
EN  GO TO 120
 135 IF INKEY$<>"" THEN  GO TO 1
35
 140 INPUT "Enter INK colour. 0-
7 "; LINE i$: IF i$="" THEN  LET
 i$="0"
 150 IF VAL i$<0 OR VAL i$>7 THE
N  GO TO 140
 155 IF INKEY$<>"" THEN  GO TO 1
55
 160 INPUT "Enter PAPER colour.
0-7 "; LINE p$: IF p$="" THEN  L
ET p$="7"
 170 IF VAL p$<0 OR VAL p$>7 THE
N  GO TO 160
 175 IF INKEY$<>"" THEN  GO TO 1
75
 180 INPUT "Enter text, 247 char
s max.       "; LINE t$
 190 IF LEN t$>247 OR LEN t$<1 T
HEN  GO TO 180
 200 LET x=VAL x$: LET y=VAL y$:
 LET i=VAL i$: LET p=VAL p$
 210 PRINT AT y,x; INK i; PAPER
p;t$
 220 PRINT #0;"Press Y to store,
 N to re-Enter "
 230 LET a$=INKEY$: IF a$="N" OR
 a$="n" THEN  GO TO 100
 240 IF a$<>"y" AND a$<>"Y" THEN
  GO TO 230
 250 LET t$=CHR$ (LEN t$+8)+CHR$
 17+CHR$ p+CHR$ 16+CHR$ i+CHR$ 2
2+CHR$ y+CHR$ x+t$
 260 FOR i=1 TO LEN t$: POKE add
r+i-1,CODE t$(i): NEXT i: LET ad
```

```
dr=addr+LEN t$: LET no=no+1
 270 CLS : PRINT #0;"Any more ?
Y or N "
 280 LET a$=INKEY$: IF a$="y" OR
 a$="Y" THEN  GO TO 100
 290 IF a$<>"n" AND a$<>"N" THEN
  GO TO 280
 300 PRINT "Number of items=";no
''"Length of data=";addr-40000''
'"Press:";AT 8,8;"1. To see text
";AT 10,8;"2. To SAVE text data"
;AT 12,8;"3. Add more text"
 310 LET a$=INKEY$: IF a$<"1" OR
 a$>"3" THEN  GO TO 310
 320 CLS : GO TO 300+VAL a$*100
 400 FOR i=1 TO no: CLS : POKE 3
9999,i: RANDOMIZE USR 40000
 410 INPUT "Press ENTER "; LINE
a$
 420 NEXT i
 430 CLS : GO TO 300
 500 SAVE "text"CODE 40000,addr-
40000
 600 GO TO 100
 999 DATA 33,50,0,9,11,10,61,254
,0,40,8,71,22,0,126,95,25,16,251
,229,62,2,205,1,22,225,6,0,78,13
,35,235,205,60,32,201
```

Machine code routine

```
9C40          0000         ORG   40000
9C40 213200   0005         LD    HL,50
9C43 09       0010         ADD   HL,BC
9C44 0B       0015         DEC   BC
9C45 0A       0020         LD    A,(BC)
9C46 3D       0025         DEC   A
9C47 FE00     0030         CP    0
9C49 2808     0035         JR    Z,PRINT
9C4B 47       0040         LD    B,A
9C4C 1600     0045         LD    D,0
9C4E 7E       0050 LOOP    LD    A,(HL)
9C4F 5F       0055         LD    E,A
9C50 19       0060         ADD   HL,DE
9C51 10FB     0065         DJNZ  LOOP
9C53 E5       0070 PRINT   PUSH  HL
9C54 3E02     0075         LD    A,2
9C56 CD0116   0080         CALL  1601H
9C59 E1       0085         POP   HL
9C5A 0600     0090         LD    B,0
9C5C 4E       0095         LD    C,(HL)
9C5D 0D       0100         DEC   C
9C5E 23       0105         INC   HL
9C5F EB       0110         EX    DE,HL
9C60 CD3C20   0115         CALL  203CH
9C63 C9       0120         RET
              0125         END
PRINT 9C53    LOOP  9C4E     #      5E3E
```

SHORT CUTS

Ray Elder with another prizewinning collection of unusual, practical or ingenious routines.

Security alert

M.A. Shortt has a similar liking as I do for 'silly' programs and this one has no real purpose except to give heart attacks to customers who press the keys of a Spectrum on the shelves of local stores!

When run the Spectrum sits there with the usual Sinclair copyright logo looking innocent and tempting, but when someone presses a key . . .

(We take no responsibility for anyone who enters it into display machines —)

```
  10 REM  alarm
  20 CLEAR 65345: FOR n=65346 TO
 65367: READ a: POKE n,a: NEXT n
  40 PRINT #0; 1982 Sinclair R
esearch Ltd"
  50 PAUSE 0: CLS : PRINT AT 10,
9; FLASH 1;" SECURITY ALERT "
  60 RANDOMIZE USR 65346
  70 DATA 58,72,92,15,15,15,30,0
,243,211,254,238,16,67,16,254,29
,32,246,251,24,234
```

Security Alert

Hurrah for Currah

The Currah microspeech unit is one of the most fascinating devices you can add to your Spectrum, but it is not easy to program. A. Welsh's program **Word Generator** will provide welcome assistance for owners of the beast by enabling easy manipulation of allophones to enable you to create a tape or microdrive dictionary, change the lines 330 and 340 for cassette storage.

Word Generator

```
   1 BORDER 1: INK 7: PAPER 1: C
LS : DIM d$(250,20): POKE 23658,
8
  10 LET keys=0: LET j=0: LET z=
0
  20 LET t$="": LET i=0: CLS : P
RINT AT 1,10;"WORD GENERATOR"
  30 FOR x=0 TO 31 STEP 8
  40 FOR y=3 TO 17: READ z$: LET
 i=i+1
  45 IF x>0 THEN  LET z=-2
  46 IF x>23 THEN  LET z=-1
  50 PRINT AT y,x+z;i;TAB x+z+3;
z$
  60 NEXT y
  70 NEXT x
  71 RESTORE
  75 FOR a=1 TO 250
  80 INPUT "ALLOPHONE No. (I-pre
fix for     intonation or E to e
nd) ";b$
  82 IF b$="" THEN  GO TO 80
  85 IF b$(1)="I" THEN  RESTORE
9030: FOR i=1 TO VAL b$(2 TO ):
READ z$: NEXT i: RESTORE : GO TO
 100
  90 IF b$(1)="E" THEN  GO TO 12
0
  91 IF CODE b$>64 THEN  GO TO 8
0
  95 FOR i=1 TO VAL b$: READ z$:
 NEXT i: RESTORE
 100 IF LEN t$<=28 THEN  LET t$=
t$+z$: PRINT AT 19,0;t$
 110 NEXT a
 120 LET keys=0: PRINT AT 19,0;t
$: LET s$=t$: PAUSE 1
 130 PRINT #0;"REPEAT,STORE,MEMO
RY,PRINT,TAPE, NEWWORD,QUIT(R,S,
M,P,T,N,Q)"
 135 LET f$=INKEY$
 140 IF f$="R" THEN  LET s$=t$:
PAUSE 1
 150 IF f$="S" THEN  LET d$(j+1)
=t$: LET j=j+1: PRINT AT 19,0;"S
TORED                          "
 160 IF f$="M" THEN  FOR g=1 TO
250: LET s$=d$(g): PAUSE 1: IF C
ODE d$(g)(1)>32 THEN  NEXT g
 170 IF f$="P" THEN  FOR g=1 TO
250: LPRINT d$(g): IF CODE d$(g)
(1)>32 THEN  NEXT g
 175 IF f$="T" THEN  GO SUB 300
 180 IF f$="Q" THEN  STOP
 190 IF f$="N" THEN  LET t$="":
PRINT AT 19,0;"
                  ": GO TO 75
 200 GO TO 135
 300 INPUT "FILENAME ";l$
 310 PRINT #0;"SAVE or LOAD (S o
r L)"
 320 LET f$=INKEY$
 330 IF f$="S" THEN  SAVE *"m";1
;l$ DATA d$(): GO TO 75
 340 IF f$="L" THEN  LOAD *"m";1
;l$ DATA d$(): GO TO 75
 350 GO TO 320
9000 DATA "a","b","c","d","e","f
","g","h","i","j","k","l","m","n
","o","p","r","s","t","u","v","w
","y","z"
9010 DATA "(aa)","(ee)","(ii)","
(oo)","(bb)","(dd)","(gg)","(ggg
)","(hh)","(ll)","(nn)","(rr)","
(tt)","(yy)","(ar)","(aer)","(ch
)","(ck)","(ear)","(eh)","(er)",
"(err)","(ng)","(or)","(ou)","(o
uu)"
9020 DATA "(ow)","(oy)","(sh)","
(th)","(dth)","(uh)","(wh)","(zh
)"," ","."
9030 DATA "A","B","C","D","E","F
","G","H","I","J","K","L","M","N
","O","P","R","S","T","U","V","W
","Y","Z"
9040 DATA "(AA)","(EE)","(II)","
(OO)","(BB)","(DD)","(GG)","(GGG
)","(HH)","(LL)","(NN)","(RR)","
(TT)","(YY)","(AR)","(AER)","(CH
)","(CK)","(EAR)","(EH)","(ER)",
"(ERR)","(NG)","(OR)","(OU)","(O
UU)"
9050 DATA "(OW)","(OY)","(SH)","
(TH)","(DTH)","(UH)","(WH)","(ZH
)"," ","."
```

```
                WORD GENERATOR

 1  a     16  p       31  (gg)    46  (err)
 2  b     17  r       32  (ggg)   47  (ng)
 3  c     18  s       33  (hh)    48  (or)
 4  d     19  t       34  (ll)    49  (ou)
 5  e     20  u       35  (nn)    50  (ouu)
 6  f     21  v       36  (rr)    51  (ow)
 7  g     22  w       37  (tt)    52  (oy)
 8  h     23  y       38  (yy)    53  (sh)
 9  i     24  z       39  (ar)    54  (th)
10  j     25  (aa)    40  (aer)   55  (dth)
11  k     26  (ee)    41  (ch)    56  (uh)
12  l     27  (ii)    42  (ck)    57  (wh)
13  m     28  (oo)    43  (ear)   58  (zh)
14  n     29  (bb)    44  (eh)    59
15  o     30  (dd)    45  (er)    60
```

Sound Synth

Robert Glavas makes it a hat trick with this, his third published program in Short Cuts . . .

A great little program which could keep you happy for hours, the machine code reads sound data from address 40000 and each number represents a 'pause' which controls the output. This table of data must end with a value 0.

There are a few demos to get you started but the best sounds will be achieved by experimenting.

Sound Synth

```
   1 REM Sound Synth
   2 REM RUN program then type
         GO TO 100,200,300,etc.
  10 LET n=40000: RANDOMIZE n: L
ET lo=PEEK 23670: LET hi=PEEK 23
671
  20 DATA 243,33,lo,hi,126,254,0
,32,+2,251,201,71,58,72,92,230,5
6,31,31,31,211,254,16,-2,198,16,
211,254,35,24,-27
  30 FOR f=60000 TO 60030: READ
a: POKE f,a: NEXT f
  40
 100 REM demo1, alien
 110 LET n=40000
 120 FOR f=250 TO 240 STEP -.05
 130 POKE n,INT f: LET n=n+1
 140 NEXT f: POKE n,0
 150 PRINT "press a key."
 160 PAUSE 0: LET l=USR 60000: G
O TO 160
 190
 200 REM demo2, spooky
 210 LET n=40000
 220 FOR f=1 TO 3: FOR s=150 TO
220: POKE n,s: LET n=n+1: NEXT s
 230 FOR s=220 TO 150 STEP -1: P
OKE n,s: LET n=n+1: NEXT s
 240 NEXT f: POKE n,0
 250 PRINT "press a key"
 260 PAUSE 0: LET l=USR 60000: G
O TO 260
 290
 300 REM demo3, hazy note
 310 LET n=40000
 320 FOR f=1 TO 200: POKE n,(RND
*15)+100: LET n=n+1
 330 NEXT f: POKE n,0
 340 PRINT "press a key"
 350 PAUSE 0: LET l=USR 60000: G
O TO 350
 390
 400 REM demo4, photon fire
 410 LET n=40000
 420 FOR s=1 TO 3: FOR f=1 TO 25
5 STEP 3: POKE n,f: POKE n+1,(IN
T f/2)+1: POKE n+2,150: LET n=n+
3
 430 NEXT f: NEXT s: POKE n,0
 440 PRINT "press a key"
 450 PAUSE 0: LET l=USR 60000: G
O TO 450
 490
 500 REM demo5, squeeky
 510 LET n=40000
 520 FOR f=130 TO 90 STEP -1: PO
KE n,f: LET n=n+1: NEXT f
 530 FOR s=1 TO 60: POKE n,f: LE
T n=n+1: NEXT s
 540 FOR f=f TO f+40 STEP 2: POK
E n,f: LET n=n+1: NEXT f
 550 FOR f=f TO f-40 STEP -2: PO
KE n,f: LET n=n+1: NEXT f
 560 POKE n,0: PRINT "press a ke
y"
 570 PAUSE 0: LET l=USR 60000: G
O TO 570
```

Daisywheel Pictures

Two very short programs which perform a unique feat, producing graphic style pictures on a daisywheel printer. Actually any printer could produce this type of pic. and it could easily be modified to operate via the Interface 1 RS232 interface on a serial printer.

The author of this minor miracle is Charles Barron who lives in Aberdeenshire, he explains:

"The copy routines normally looks at each pixel on the screen and prints a dot if it is coloured (INK) and leaves it blank if it is not (PAPER) coloured. My routine prints a * instead of a dot, this means of course that it is also eight times the usual size!"

"First of all give the printer a command to print in as compressed a mode as possible so the *'s are as close together as possible, also use the least line spacing you can on your printer, possibly 8 or 10 lines per inch instead of the usual 6."

"The routine prints sideways and needs to get 176 characters per line, if you cannot do this with your printer then you will have to settle for only printing part of the screen. This can be calculated by taking the maximum CPL your printer is capable of from 175 and replacing the 0 in line 60 with the result."

"Finally switch off the automatic perforation skip if it is enabled on your printer, and then, at last, load and run the program and have your screen picture ready on tape for the program to load and then print."

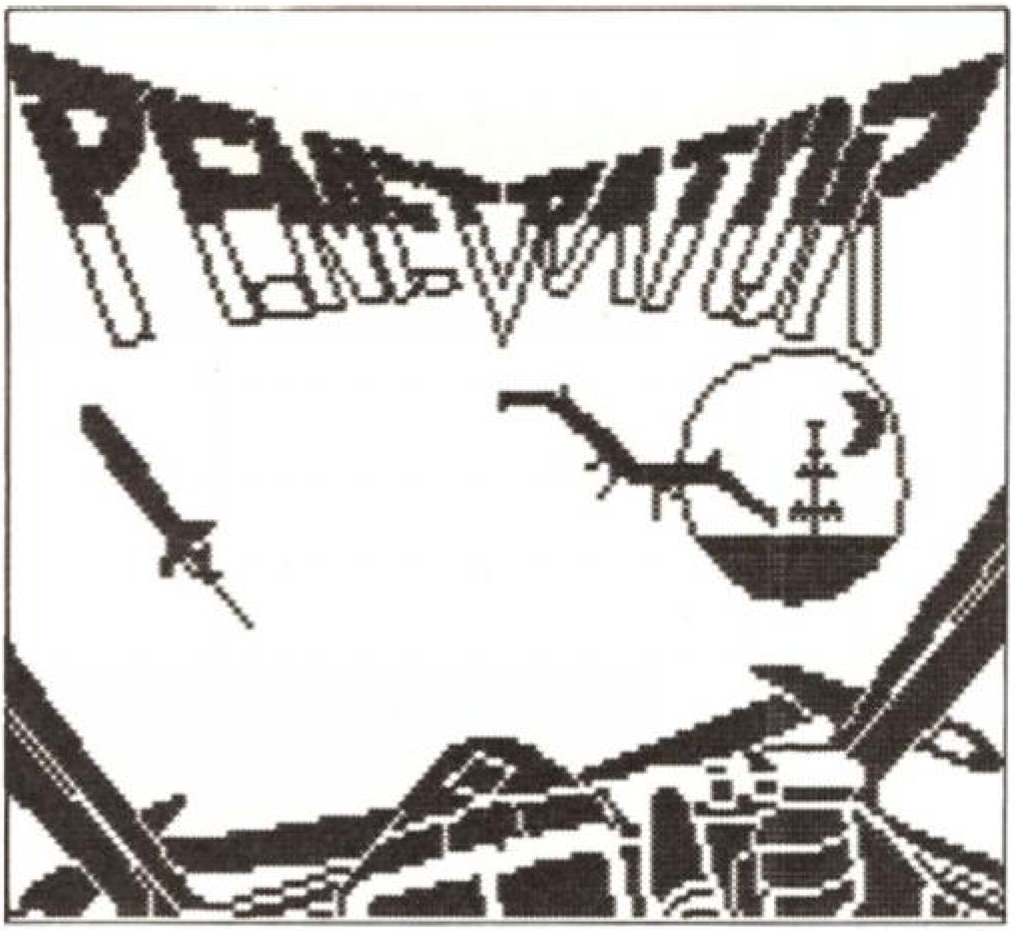
See Figure 1 for example. These clever printer routines win Charles Barron the Star Cut award for this month.

```
   1 REM Daisywheel graphics
  10 REM alter linespace
  20 LPRINT CHR$ 27;CHR$ 158;CHR
$ 2
  30 REM alter chars per line
  40 LPRINT CHR$ 27;CHR$ 159;CHR
$ 7
  50 FOR g=255 TO 0 STEP -1
  60 FOR f=175 TO 0 STEP -1
  70 IF POINT (g,f) THEN  LPRINT
 "*";: GO TO 90
  80 LPRINT " ";
  90 NEXT f
 100 REM newline
 110 LPRINT CHR$ 10
 120 NEXT g
```

```
   5 REM Daisywheel Copy
  10 FOR x=0 TO 21
  20 FOR y=0 TO 31
  30 LPRINT SCREEN$ (x,y);
  40 IF CODE SCREEN$ (x,y)<165 A
ND CODE SCREEN$ (x,y)>143 THEN
LPRINT "*";
  50 IF CODE SCREEN$ (x,y)=0 THE
N  LPRINT "*";
  60 NEXT y
  70 LPRINT CHR$ 10
  80 NEXT x
```

Daisywheel Pictures

Figure 1



UDG Grids

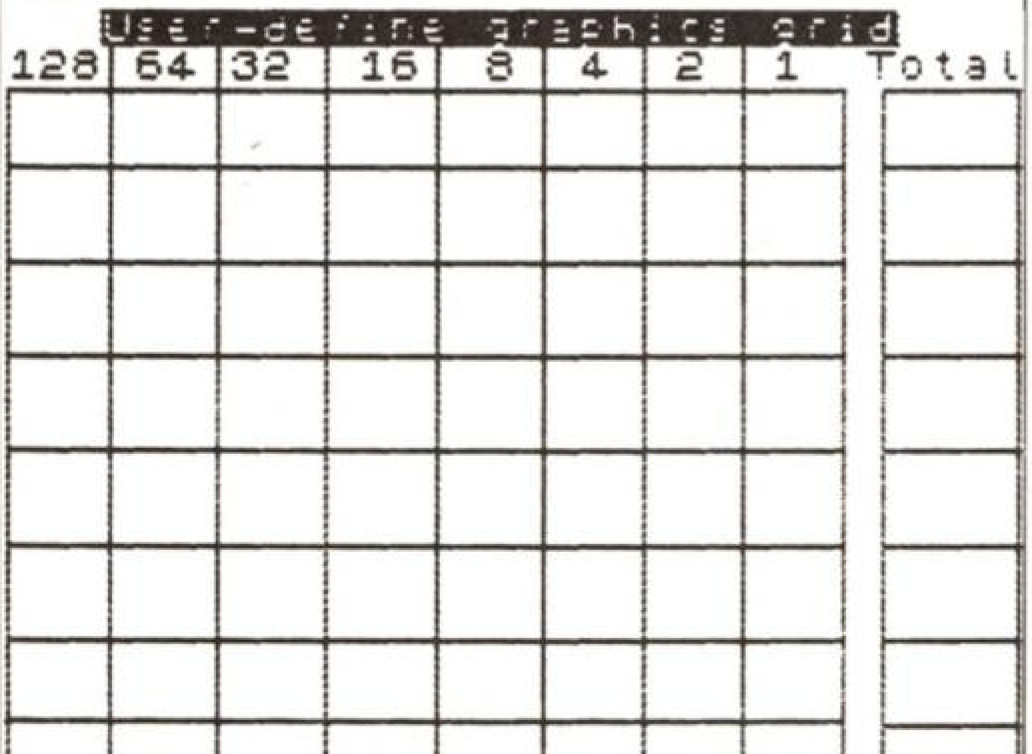
We haven't printed one of these for many moons now and we know that we have quite a few new readers who would find it useful. Mr. J Plater of Kent sent n this rather neat version and they are great for those off computer planning sessions (Figure 2).

Designed for use with the ZX and Alphacom printers it'll keep churning out the grids until BREAK is pressed.

UDG Grids

```
   5 REM uDG Grids
  10 INVERSE 1: PRINT AT 1,3;"Us
er Defined graphics grid"
  20 INVERSE 0: PRINT AT 2,0;"12
8 64 32  16  8  4  2  1  Total"
  40 PLOT 0,0: DRAW 0,150: DRAW
210,0: DRAW 0,-150: DRAW -210,0
  50 PLOT 220,150: DRAW 34,-0: D
RAW 0,-150: DRAW -34,0: DRAW 0,1
50
  60 PLOT 26,0: DRAW 0,159: PLOT
 53,0: DRAW 0,159: PLOT 80,0: DR
AW 0,159: PLOT 108,0: DRAW 0,159
: PLOT 135,0: DRAW 0,159: PLOT 1
60,0: DRAW 0,159: PLOT 185,0: DR
AW 0,159
  70 PLOT 1,17: DRAW 208,0: PLOT
 1,34: DRAW 208,0: PLOT 1,54: DR
AW 208,0: PLOT 1,74: DRAW 208,0:
 PLOT 1,94: DRAW 208,0: PLOT 1,1
14: DRAW 208,0: PLOT 1,134: DRAW
 208,0
  80 PLOT 221,134: DRAW 33,0: PL
OT 221,114: DRAW 33,0: PLOT 221,
94: DRAW 33,0: PLOT 221,74: DRAW
 33,0: PLOT 221,54: DRAW 33,0: P
LOT 221,34: DRAW 33,0: PLOT 221,
16: DRAW 33,0
  90 COPY
 100 PAUSE 1: GO TO 10
```

Figure 2



Poster Print

```
  10 REM SPECTRUM 128/SERIAL 805
6 PRINTER DUMP PROGRAM
  20 IF PEEK 60000<>62 THEN  GO
SUB 250
  30 FORMAT "p";1200
  40 LOAD !"DUMP"SCREEN$
  50 LET p$=CHR$ 24+CHR$ 20+CHR$
 15+CHR$ 27+CHR$ 48
  60 GO SUB 200
  70 LET st=0: LET fi=135: GO SU
B 130
  80 LPRINT "''''": LET p$="": FO
R a=1 TO 40: LET p$=p$+CHR$ 16+C
HR$ 17: NEXT a: GO SUB 200: LPRI
NT "''''"
  90 LET st=136: LET fi=255: GO
SUB 130
 100 LET p$=CHR$ 18+CHR$ 27+CHR$
 50+CHR$ 13+CHR$ 10
 110 GO SUB 200
 120 STOP
 130 FOR y=175 TO 0 STEP -1: LET
 p$="": FOR x=st TO fi
 140 IF POINT (x,y)=0 THEN  LET
p$=p$+" ": NEXT x: GO TO 160
 150 LET p$=p$+CHR$ 8: NEXT x
 160 LET p$=p$+CHR$ 10+CHR$ 13
 170 GO SUB 200
 180 NEXT y
 190 RETURN
 200 FOR a=1 TO LEN p$
 210 POKE 60001,CODE p$(a)
 220 RANDOMIZE USR 60000
 230 NEXT a
 240 RETURN
 250 LET t=0: FOR a=60000 TO 600
10: READ n: POKE a,n: LET t=t+n+
a: NEXT a
 260 IF t<>660944 THEN  CLS : PR
INT  FLASH 1;"ERROR IN DATA LINE
 9020.": "C#g))","E#b))","G#d))
"
 270 DATA 62,0,205,0,91,205,39,1
,195,0,91
 280 RETURN
9999 PAUSE 200: POKE 23736,181::
 SAVE "P.DUMP" LINE 0: GO TO 999
9
```

Poster Print

John Scott of Stroud sent us a huge screen dump to demonstrate his Spectrum Serial 8056 printer routine. To use it load in the screen picture you want to print and save it to RAMdisk with: SAVE !"DUMP" SCREEN$

Now load your saved version of his program, if you saved it using the autorun line number then you can go away and forget it for half an hour as it takes that long to print it.

The print it done in two 'halves' sideways on the paper and once completed then you will have to sellotape the two halves together to produce a unique wall poster, especially if you used a graphics program to create an original picture.

SHORT CUTS

WAR GAMES

CCS has released three new challenges for Spectrum Generals. Tony Hetherington reports from the front.

CCS is rapidly achieving the reputation as being **the** wargamers. Three new games as different as the campaigns they simulate add to an impressive line up which already includes games such as Desert Rats and Arnhem.

In this latest batch of blood 'n guts for the thinking gamer, prospective Generals play Napoleon at the battle of Eylau, replay the disastrous WW1 Gallipoli campaign and tangle with monsters and wizards in Swords of Bane.

Gallipoli

The Gallipoli campaign was typical of the disastrous WW1 conflicts in which both sides suffered terrible casualties





Gallipoli

largely due to idiotic Generals. Would it be different with you in charge?

Gallipoli can be played by one, two or three players each controlling the British, ANZAC (British, Australian and Gurkha forces) and Turkish armies with the computer ready to play the game if you haven't enough players.

The object of the game seems relatively simple in that the Allies with a total of 60,000 troops must advance to take the Turks main ammunition dump before reinforcements arrive to support the 22,000 Turkish troops.

The gameplay follows accepted wargame standards with a cursor being moved around the forces issuing commands. It is the choice of orders that make this game unique.

As well as the usual move and fire orders Gallipoli troops can dig trenches forming the almost impassable lines typical of WW1.

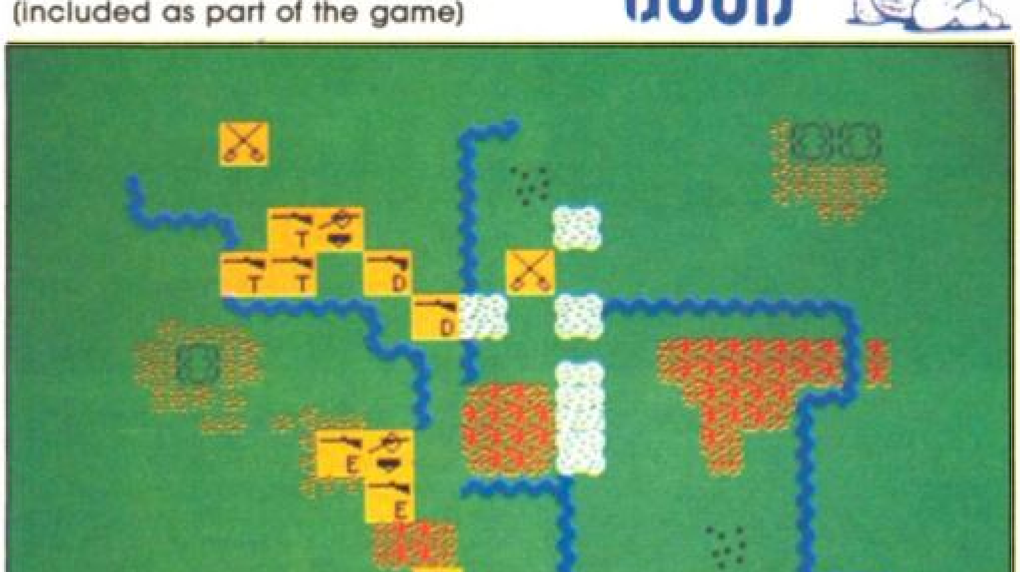
In the expanded 128K version (included as part of the game) players can dig tunnels between trenches and set bombs off under the enemy forces to try to break the deadlock.

The expanded version also extends the combat sequence from the standard comparison of opposing armies strengths and terrain cover by the inclusion of a sub game.

You can play a single soldier looking out onto a battlefield that contains 10 hidden enemy soldiers. These fire at you in turn revealing their position giving you a few seconds to shoot at them. After twenty shots the computer evaluates your performance which determines the result of all battles in that turn.

Unfortunately this spoils the game reducing a challenging simulation into a fairground shooting gallery. Luckily it's only an option that can be ignored.





Napoleon at War

Napoleon at War

The battle of Eylau was close run affair between Napoleon and the combined Russian and Prussian forces and therefore ideally suited to conversion into a wargame.

This simulation offers prospective Napoleons not only a computerised opponent but also digital commanders that will carry out your general orders as best as they can.

If they run out of ideas or feel their position is impossible then they'll send you a message.

Alternatively you can run the whole battle yourself as in a standard wargame but I preferred to use the Commanders which added a new dimension to a fascinating period of history.

The game itself is a tricky ballancing act between holding onto the town of Eylau while striking at the advancing enemy. Sit back and wait for the attack and you'll be overrun but attack too much and you'll be left with no defense.

Napoleon at War definitely captures the atmosphere of the age of horse, cannon and musket.



Swords of Bane

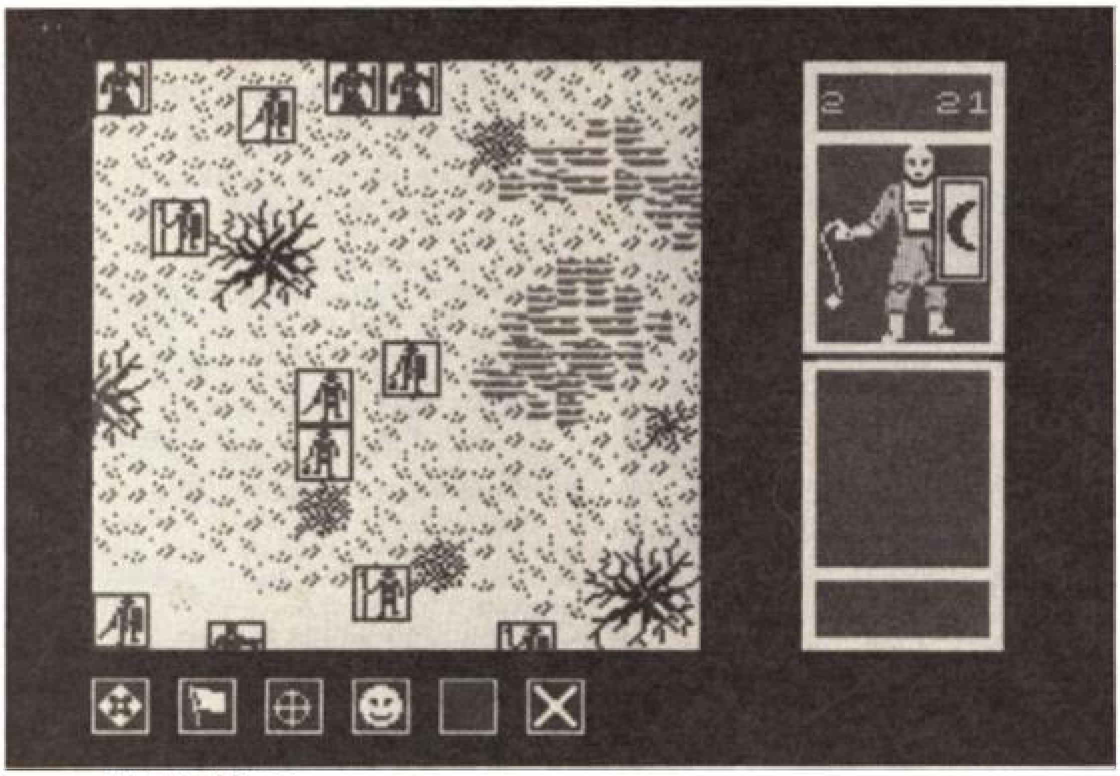
As a change from recreating historical battles here's a chance to fight monsters with wizards in the first computerised fantasy wargame.

A formidable army of earth and water elementals and demons are approaching the village led by the Fire Demon. You must raise an army of warriors and wizards to stop them.

By spending your restricted resources you can raise an army consisting of warriors armed with maces, swords, spears and crossbows as well as wizards ready to wield their magic.

Unfortunatley you meagre purse restricts your choice somewhere between a small elite core of wizards to a large peasant rabble.

This is no ordinary wargame. The wizards can hurl spells from quite a distance and the monsters drain the life energy from those that get too close and so you must arrange your forces so that when you strike, you kill, otherwise you'll actually make the enemy stronger!



Swords of Bane

Again, 128K owners get a little extra with the inclusion of two more scenarios that take the battle into a forest and finally to an inn.

Swords of Bane is an attempt to recreate the depth and tremendous possibilities of fantasy wargaming which hasn't quite worked.

There's not the variety of monster type of characteristics that such a game should have or the selection of spells to set a wizard apart from a bowman. The result is a game that will be ignored by pure wargamers and will disappoint fantasy freaks.



GRIM



Conclusion

Napoleon at War is my pick of the three games. It's relative simplicity will apeal to beginners who can take control from their computer Commanders as they feel ready then there's a further two levels to challenge the best.

Gallipoli is a more complex game and reflects the painfully slow progress of WW1 campaigns (as compared to the free flowing Napoleonic battles which is why it's one of my favourite wargaming periods). The number of troops involved and the tactics required to gain any ground at all saves this one for the experts. The sub game actually spoils the game but can be avoided.

Finally Swords of Bane just didn't work. A good idea was there but it was smothered in lack of depth, variety and clumsy control system that had you ordering troops on a fraction of the battlefield without being able to see the rest.

Gallipoli and Napoleon at War cost £8.95. Swords of Bane £7.95.

CROSSFIRE

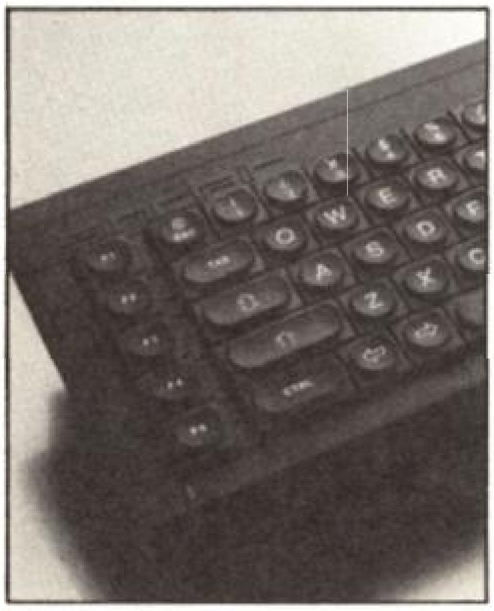
QL critic

Dear Sir,
I am a QL owner and I should not say this, but to be absolutely honest the QL should die, and this is because of QDOS. QDOS is a poor programming language. Of course it is 100 times better than microsoft basic and it beats Texas Instruments implementation of BASIC any day. However, compared to good old Sinclair Basic, QDOS is the pits. They blew it when they attempted to bring the Sinclair Basic more in line with common business basic to appeal to the business community.

There is no device in existence, including the mouse, trackball, touch sensitive screen, or any other method of computer/human interfacing that is anywhere near as effective or user friendly as Sinclair Basic, with its one touch entry of keywords and automatic syntax checking.

I and very many other Americans would never have bothered to learn to program if it were not for Sinclair Basic. In fact 90% of the people I know who can program, originally learned on a Sinclair machine. If they (the QDOS writers) wanted to add extra features with the QL they should have done so, leaving the kernal of Sinclair Basic intact. Whoever wrote QDOS took a good thing and murdered it. These people are directly responsible for the QL's demise. If he or she was so het up on structured programming he should have offered a Pascal or some other structured language ROM as a plug in option.

Ullyses B. Adams, Philadelphia, U.S.A.



Pen Pals

I am involved in volunteer work with children and I also use my Spectrum 48K for pubs and stat keeping. Anyone interested in swapping educational and other useful programs?

Write to: Jeff McMeel, IPO 5371 Tokyo, -100-31, Japan.

I am French and perhaps the only owner of a Beta Plus disc (double density) drive in my country. I would very much like to correspond with ZX readers on computing matters. In France the magazines on computers are not very good, so I prefer ZX.

Mr. Kusbacs, 7 Ave de Isle de France, 95300, Pontoise, France.

Omnicalc

Dear Sir,
Is there a book available which could show how to vary the width of the columns in Omnicalc 2? This is an excellent spreadsheet for my Spectrum 48K but there are times the amounts to be entered have more than 7 digits allowed per column.

Also, there are times I need to enter a line of text.

David A. Vail, Lindheim, We. Germany.

MINDPLAY

Peter Sweasey, our surly sentinel at the portal of the adventure market with the pre-Christmas releases.



The sound of carols drifts through the grating: "... tis the season to be jolly..." It bloomin' well is not! It's Christmas and I'm still down here in the damp dungeons, no fairy lights to brighten the dark, no decorations to make life less dismal. Not even a mince pie. And Bryan, my editor, must be a direct relative of Scrooge. Honestly, if I ... (stop whingeing and get to work, or it's the Orville Christmas Special for you — Ed).

Actually, it's not too bad down here this month, since the adventures for review are mostly pretty good (I could have called them "a bunch of real crackers" but I like to think I have a little integrity!). Mosaic provide two high quality games based round unusual licensing deals — good to see a company steering away from big name films and television programmes, which tend to produce boring games. And who would have thought, this time last year, that CRL would be one of the most successful adventure producers of '86. Another above average spoof from them, this time via Saint Brides.

To be honest, us computer journalists are forced to cheat and write our stuff several weeks before you read this — in fact fireworks are making irritating noises while I write this. Between now and when this issue is on sale, a whole load of other potentially perfect presents for adventure nuts may be released, including Delta 4's Colour Of Magic, Melbourne House's Dodgy Geezers, Oink! from CRL, revamped Silicon Dreams from Rainbird, Domark's Live and Let Die and possibly many more. Plenty to look forward to, I'm sure yule agree. (Damn! There goes the integrity!). But will any of them arrive in time?

That question will be answered in due course: in the meantime, on with the reviews.

Bugsy

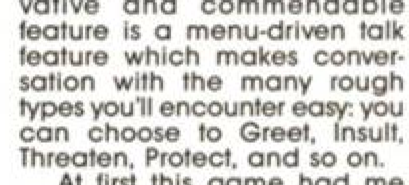
CRL/Saint Brides
£7.95

Da broads from Saint Brides are back, wit' a tale about a ganshter who wants ta woik his way up from penniless to Public Enemy Number 1. "Ain't nuffin new about dat" ya say? Well this ganshter is a three foot tall pale blue rabbit wit' a cute little power puff tail. Da name is Bugsy. Bugsy Maroon. Da place is Chicago, 1922.

An' dat's ... I mean, and that's as much as I'm prepared to write in that accent. The whole of this new spoof is written in that style — Bugsy himself being somewhat short of intelligence — which provides some of the laughs. But they come from many sources; they're more frequent than in the Delta 4 adventures, and the sort of humour is more varied (from corny to, er, very corny). I won't spoil any of the jokes; suffice to say, Bugsy is always amusing and at times funny.

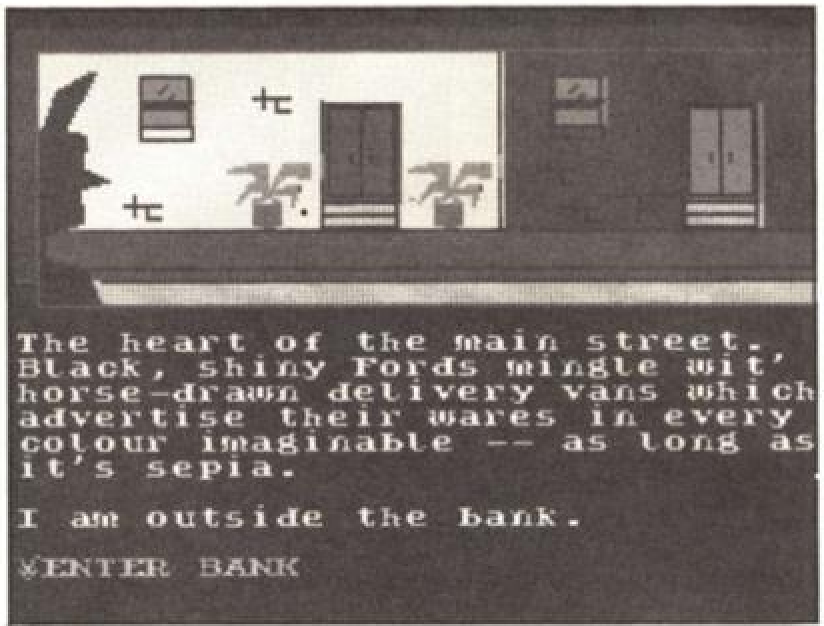
The game is expertly Quilled and in two parts. There are split screen graphics at every location, and these are pleasant enough, if slow. An innovative and commendable feature is a menu-driven talk feature which makes conversation with the many rough types you'll encounter easy: you can choose to Greet, Insult, Threaten, Protect, and so on.

At first this game had me totally stumped, but once I had been set in the right direction I found it logical and very enjoyable. Bugsy is no classic, but it is a polished piece of software which will provide plenty of entertainment for a reasonable price.





Bugsy



Bugsy

The Archers

Mosaic
£9.95

All together now: "Dum dee dum dee dum dee dum, dum dee dum dee daa daa". Yes, Mosaic have joined with Level 9 to produce a gamed based on the Radio 4 soap opera which was topping the ratings long before EastEnders.

Rather than casting you as an Archers' character — which would have led to a fairly mundane puzzle-solving game with Ambridge as a setting — you take the role of scriptwriter. There are four separate games, and you control the life of a different person in each: wealthy estate-manager Jack Woolley, spoilt young woman Elizabeth Archer, would-be country singer Eddie Grundy and ageing wine-bar owner Nelson Gabriel are the chosen four. For people unfamiliar with the show — like myself — there is a complete list of characters you may encounter, their personalities and relationships with each other.

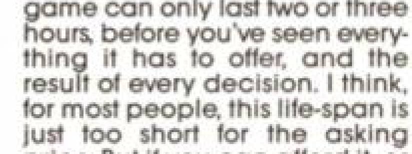
This is not a "true" text input adventure. Instead you are presented with a situation, and most choose one of three different options as to what your character could do. The selection may lead to a whole, complicated sub-plot developing, with more choices to be made; and you can indulge in outrageous story twists. Problem is, you are constantly being judged: on ratings performance, the opinion of Radio 4's controller, and listener reaction to the serial's realism and moral standards, or lack of them.



I have never listened to The Archers (must do sometime) but found myself becoming engrossed in the lives of these people. I also found myself frequently laughing out loud, which is a very rare effect for the game to have on me. I simply loved having the power to create interesting and amusing situations. I expect Archers devotees will enjoy the game even more, being able to determine the lives of their favourite characters.

The game has two main flaws. The first is that the Spectrum isn't quite sophisticated enough to handle the number of continuous plots that it's required to. The game doesn't always realise when one action makes another impossible. So immediately after I had sent Higgs the gardener to the asylum, I was given the option of using him to help Jack. There are other illogicalities. Sometimes a major plot decision is taken — like Eddy Grundy's wife leaving him — and nothing else is heard on the matter, or she suddenly appears back with him. More memory to prevent such faults would have been available by leaving out the graphics, which are down to the usual, hilariously hopeless Level 9 standard.

The other main problem is that this costs £10, yet each game can only last two or three hours, before you've seen everything it has to offer, and the result of every decision. I think, for most people, this life-span is just too short for the asking price. But if you can afford it, or you're a true Archers fan, this will provide a good deal of fun for a day or two — perhaps ideal for an older member of your family on December 25.



Prehistoric Adventure

Crusader Computing
£9.95

Find the Elixir of Youth somewhere in a strange Stone Age world which includes burger bars and yachts. Various prehistoric creatures have already drunk this elixir, which explains the evolutionary problem of why they're around at the same time as man. These creatures will have to be tackled to gain the potion.

To succeed as a new adventure company must be hard, and Crusader are to be congratulated on the quality of their packaging: a double cassette size box, and a colour dinosaur poster. Their clue sheet is very well devised. Neither have they taken the easy route by using The Quill, instead they've developed their own machine code adventure system.

However, I fail to see what the advantage of doing this was. It was not to enable advanced compression techniques: the game is text only, but there isn't a remarkable amount to be read. Input has to be two words only, which is unacceptable in this age of Level 9 style parsers. Furthermore, few words can be abbreviated, which is poor programming.

One good point of the parser is that it tells you exactly what it doesn't understand: the first, second or both words, or the particular combination. But then the vocabulary is so small that this happens annoyingly often. There's no EXAMINE. Also, I found the mixture between authentic pre-history and more modern developments made the game a mish-mash rather than "lighthearted".

All these faults would be criticised in a budget game. This is vastly overpriced at the standard maximum, £10. You can buy many better adventures for same price or less. Available mail order only from: 18 Henley Wood Road, Earley, Reading, Berks RG6 2EE.



Twice Shy

Mosaic
£9.95



It always amazes me how Dick Francis can come up with so many best sellers ALL based round intrigue in the racing world. Anyway, Twice Shy was one of his, Mosaic have released the adventure, and it's programmed by RamJam, authors of an all-time favourite of mine, Valkyrie 17.

You are Jonathan Derry, a Physics teacher who has become custodian of some computer tapes. The purpose of these is unknown to you at the start, but other people seem to know — and they'll stop at nothing to get them.

First thing you notice when you load up the adventure is the excellent screen presentation. There's a static box containing the location description and another for the graphics. The bottom half of the screen scrolls, and is used for input, messages and object reports. The graphics system is similar to Ocean's adventures: small, and not every location has its own illustration, but pictures which do appear are high resolution and pleasant to look at.

Once you begin playing, however, the true strength of this game becomes apparent. It's a wonderful adventure, packed full of things to do and look at, even if they're not relevant. So you can switch on and watch the television. Examine the flying ducks. Shoot your friends with the rifle. Or crash your car. You'd be amazed how many games I receive won't let the player do things like this. There are numerous small details which add so much. Your house is packed full of objects to find, as a real house would be.

There's a sense of humour, and other welcome features like well-used independent characters and the ability to interact with them. Vocabulary is adequate.

When you need more money during the game (you have to buy things like petrol, drinks, car repairs), you must visit the race course at Newmarket. Once there, you load *side 2 where you have six* graphically portrayed races to bet on. At your disposal are odds, racing conditions and form cards. Apart from having doubts about whether adventurers will like their problem-solving interrupted in this way, I also found it difficult to win anything. I'm not a betting man you see, and haven't the foggiest idea about half the factors I'm supposed to consider.

The instructions should have dealt with this, with some sort of "Idiots Guide to Horse Racing". But the instructions are too brief, telling no more of the plot than that I gave at the start of the review. More should have been given, including a guide to the characters you will encounter. The best thing to do, presumably, is read the book — though I haven't had the chance to do so yet, and I think it's slightly unfair of Mosaic to expect all adventurers to wade through a whole novel.

Those gripes aside, Twice Shy is technically smoother than Valkyrie 17 and seems as good in other ways. Which means another Monster Hit. A good Christmas present for someone who doesn't play many adventurers, but give them the book with it.

MONSTER HIT

HELPLINE



Suddenly last Thursday, there was a loud thud as something fell from the dungeon roof. The dust cleared and my excitement mounted as a tubby figure with a sack over his shoulder slowly became visible. Could it be?

No, it wasn't. Santa hadn't come early, instead it was the Argus Press postman, who — no doubt due to one Christmas tipple of sherry too many — had fallen through the grating. And the sack wasn't full of prezzies, but your problems. No rest for me this year . . . Still, at least the festive spirit seems to have reached many of you, since I also received several useful offers of help this month.

Most popular game this month on the helpline recently has been **Kentilla,** re-released by Mastertronic. I always found this game infuriating, with its very sluggish input system and aggravating randomness. One person who likes it, however, is Anthony Dunn from Camberley. He wants to enter Tylon's Castle. To do this, you need to untie

Timandra while still alive, then proceed to the castle, and wait for him and Zelda to arrive. When the offer is made, CLIMB UP. Thanks to Anthony for a complete solution to **Mindstone** by The Edge. Stuart Bell from County Durham asks a number of questions, the answers to which are as follows. To cross the sea, you need the oars for your boat (EXAMINE VEGETATION for a second time). PULL BOAT out to sea, then GO BOAT and ROW WEST. When it starts to flood, BAIL OUT BOAT WITH CHALICE. PUT DIAMONDS INTO CRUCIBLE in the laboratory, PUT CRUCIBLE INTO FURNACE, TURN DIAL TO ON then TURN DIAL TO OFF and remove the crucible again. To open the chest you must dip the gold key into the green fluid in the laboratory, but make sure you take your ring off first. Kill the rattling quarg before carrying it across the River Cara.

John Hunter cannot see the Ward of Disintegration which he wishes to kill. You need to wear the gold ring. The troll can be killed with the crystal. John also provides the answers for some of the pleas I printed in the November issue, for which I thank him, and he asks for help with Madarin's **Time Of The End.** STROKE ANIMAL in the Oasis; and when back in the Alien Lab, use the robot to fix equipment.

Paul Newport wants to lower the chandelier in **Sorcerer Of Claymoregue Castle.** Go to the Plain Room and PUSH WEST wall, which will enable you to find the Methuselah and unravel spells. Go to the Ballroom, CAST UNRAVEL, leave quickly and don't go back until you hear the glass plummet.

Next up, some humorous games, starting with the "wacky" **Quest For The Holy Grail.** An adventure whose name I cannot decipher, from South Africa, does not know what to do once he has found the artefact in question. You must return with it to the Throne Room at Camelot and PUT GRAIL ON THRONE. Avoid the sorcerer on your way out of the caves.

Andrew Neville wants a job in Melbourne House's **Hampstead**. I presume you can reach the train, and have the bracket from the industrial estate. Take and read the card you are offered, GIVE BRACKET to Justin and take and wear his tie in return. Buy, take and wear a suit from the tailors. Go to the Gentleman's Club, GIVE CARD and say YES to what's offered. Thanks for your Sinbad solution Andrew.

Very big problems

The girls from Saint Brides have managed to flummox Jamie Ogden with their **Very Big Cave Adventure.** To cross the chasm is a tortuous process. Go to the Wellie house and GET LOG. SAY COMM to be transported back to the Debris Room. You cannot carry both the log and the lamp. So provided you left the lamp in the Debris Room, DROP LOG, GET LAMP and carry it one location west. DROP LAMP and return for the log. Although you won't be able to see anything, you can still pick it up and take it west. Then repeat this process of swapping objects until the chasm, where you can drop the log to form a bridge. And yes, the space invaders are of use. Defeat them then EXAMINE CHARACTERS. You'll find a mothership. OPEN AIRLOCK of this to find a treasure. All treasures should be dropped in the Wellie House.

Ian Gilfillan is stuck in Delta 4's **Bored Of The Rings.** "I cannot find a way from the forest, except by arriving at this willow tree that eats me. Is there another way?" No. Let the tree eat you. Then CRY HELP. And in **The Hobbit** which Ian also asked about, you need to wear the ring to successfully escape in the barrel. It keeps slipping off, so check carefully.

And finally, Nick Bailey of Addlestone is unable to leave the first few locations in Interceptor's **Aftershock.** You need the chair from your office. Take it to the lift, EXAMINE LIFT, CLIMB ONTO CHAIR, REMOVE what you found on examination and CLIMB OUT OF LIFT.

Many thanks as ever to John Wilson of Rochdale.

Write to me

Grandmother is asleep in front of the Queen's speech on telly. Young Johnny has already broken Jane's doll with his He-Man. The radio-alarm clock won't work and you were given five last year anyway. The place is covered with wrapping paper and, to cap it all, you can't get out of the first location of the bloomin' adventure which Aunty Deirdre bought you. You've been trying it since Johnny woke you up at 4 am in the morning and it has definitely put paid to any ideas of 'peace to all men' which you may have read.

But — help is at hand. You don't have to write as far as Greenland this time. And it can be done quicker than it takes to re-heat a mince pie. Just fill in the coupon here. Or if there's a lot you need solved, send a letter.

A few rules: British correspondents, please enclose a stamped, addressed envelope if you want a personal reply rather than wait for the magazine to come out. If you are writing from abroad, just enclose an envelope — I'll add the postage. I try to respond within two months but I can take longer (on the other hand, you might receive an immediate reply). I ONLY DEAL WITH ADVENTURES. Not arcade games; nor technical problems (write to Crosswires about those); nor arcade adventures (Gargoyle games included). Finally, please put the name of the game you're writing about on the back of the envelope.

Send all coupons, offers of help, Christmas cake, unwanted presents, cards and tinsel to: Mindplay, Dismally Undecorated Dungeons, ZX Computing, 1 Golden Square, London W1R 3AB. Hope you all have a fun time. See you in '87!

Title: ....

Company: ....

Problem: ....

....

....

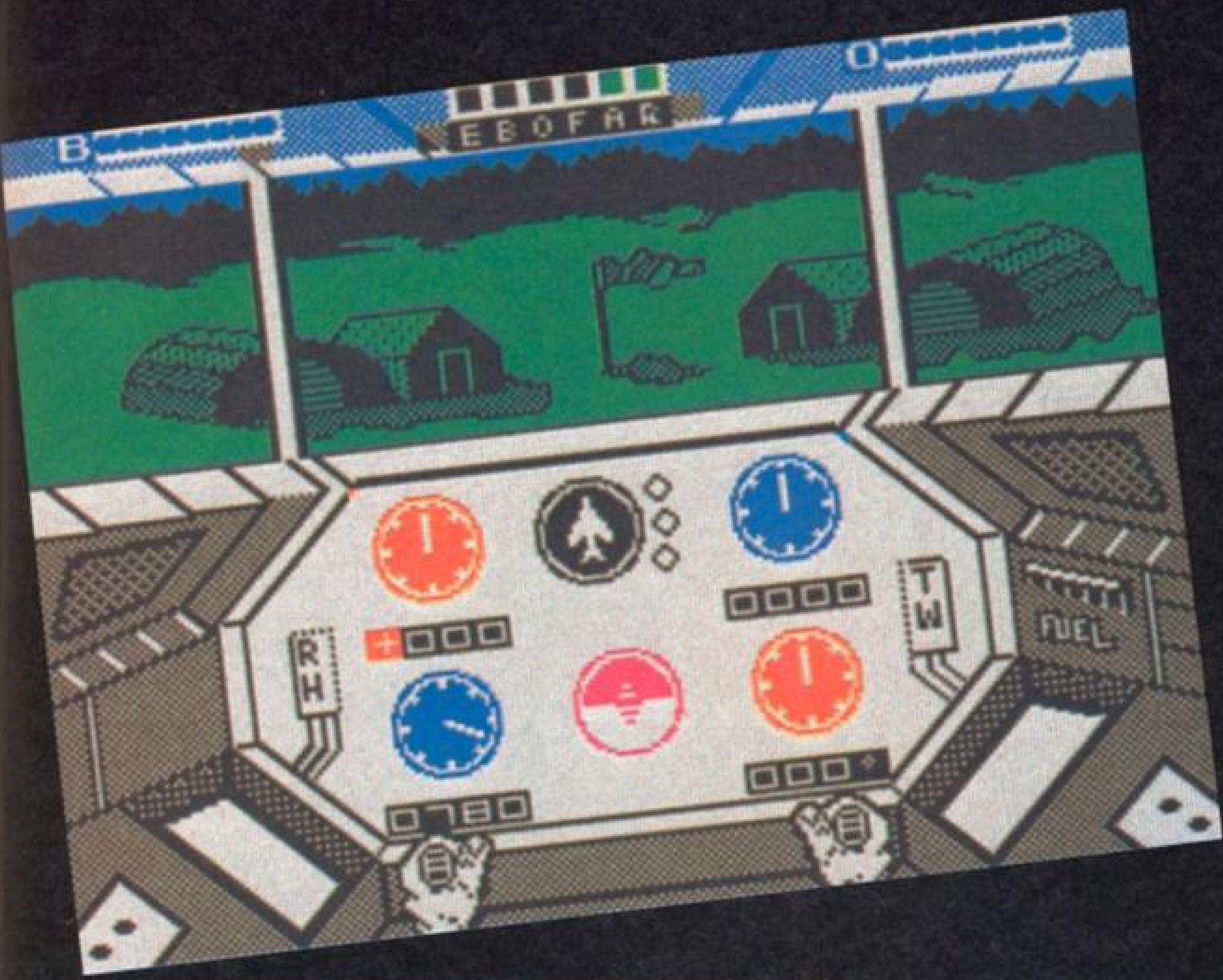
I can help solve: ....

Name: ....

Address: ....

....

INFILTRATOR



Infiltrator
US Gold
£9.95

Forget Batman, Bond and the rest of the everyday heroes. In US Gold's Infiltrator you ARE Johnny "Jimbo Baby" McGibbets!

Latest in the line of McGibbets superstars your achievements are legendary as a neurosurgeon, sportsman, diplomat, pilot, TV star, motorbike racer, film megastar and all round nice guy with a nifty haircut and designer bulletproof jeans.

It comes as no surprise to you that the world needs you to save it again. This time it's the Mad Leader up to his old tricks as he's threatening to extinguish all life and being generally unpleasant by blackmailing the world leaders. They hang up and call you instead.

As the world famous "Infiltrator" this shouldn't be too much trouble especially since you've just got your new Whizzbang Enterprises Gizmo ADHX-1 Attack Helicopter known affectionately as the "Snuffmaster".

The game begins with you sitting inside the Gizmo waiting to trigger the Whizzbang Whirler engine complete with Whizzbang Whomper turbo booster capable of a top speed of 900 knots.

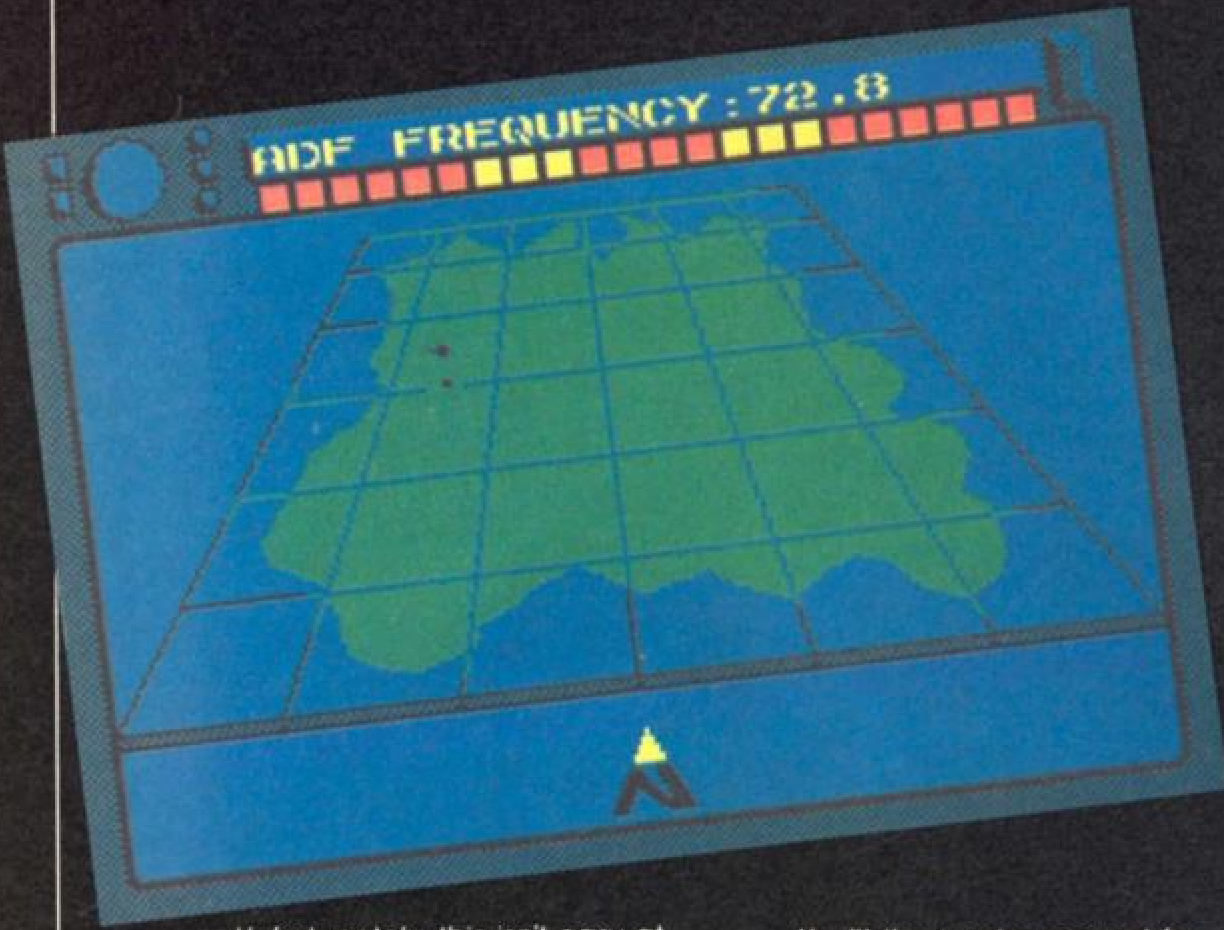
At your finger tips are the controls for the Gizmo's automatic direction finder (why should heroes navigate?), communications system, two Whizzbang Whizzer 20mm cannons and four Whizzbang Waster air to air missiles.

The screen shows the Gizmo's controls and Johnny's hands that mimic your joystick moves. Press the fire button and the on screen thumb does the same. At last a game featuring "Thumbovision"!

Your first mission is to fly secretly to the Mad Leader's base, infiltrate it and photograph the war plans.

This involves a hazardous flight over enemy airspace in a Gizmo that doesn't want to fly in a straight line. Punching up the navigation screen will get you the co-ordinates that are then fed into the ADF. Then it should be a simple task of keeping the ADF indicator level.





Unfortunately, this isn't easy at all. In fact it's extremely difficult.

The Gizmo seems to have a life of its own and heads off in the opposite direction at the most awkward moments. For example when a plane flies into view.

You must act quickly as all planes look the same and since you're supposed to be the good guy you can't just blast anybody. Instead you must hit the communicator and send a request for the plane's ID.

You'll then get a request for your ID but with a call sign such as Seth, Rhambow or Scum. These are obviously some of the Mad Leader's deranged followers so you can con them by sending the reply OVERLORD. INFILTRATOR is your real ID which of course you send to friends.

Send the wrong ID or none at all and you're in a dogfight to the death.

The enemy will fire either heat seeking or homing missiles which you can deflect if you can launch chaff or flares in time. Then it's up to your cannons or missiles to settle the argument.

Should you still have enough fuel and haven't been shot down you can land secretly just outside the enemy's base.

Inside the base the game switches from being a flight simulator to a commando style game as you try and duck past the guards to search the camp for the plans.

All purpose hero

Naturally you come prepared with your all purpose hero's kit complete with false papers, sleeping gas for troublesome guards, a mine detector, camera, explosives and gas grenades.

Inside the buildings you can search the cabinets, cupboards and safes for the documents as well as useful objects such as security passes and a less conspicuous change of clothes.

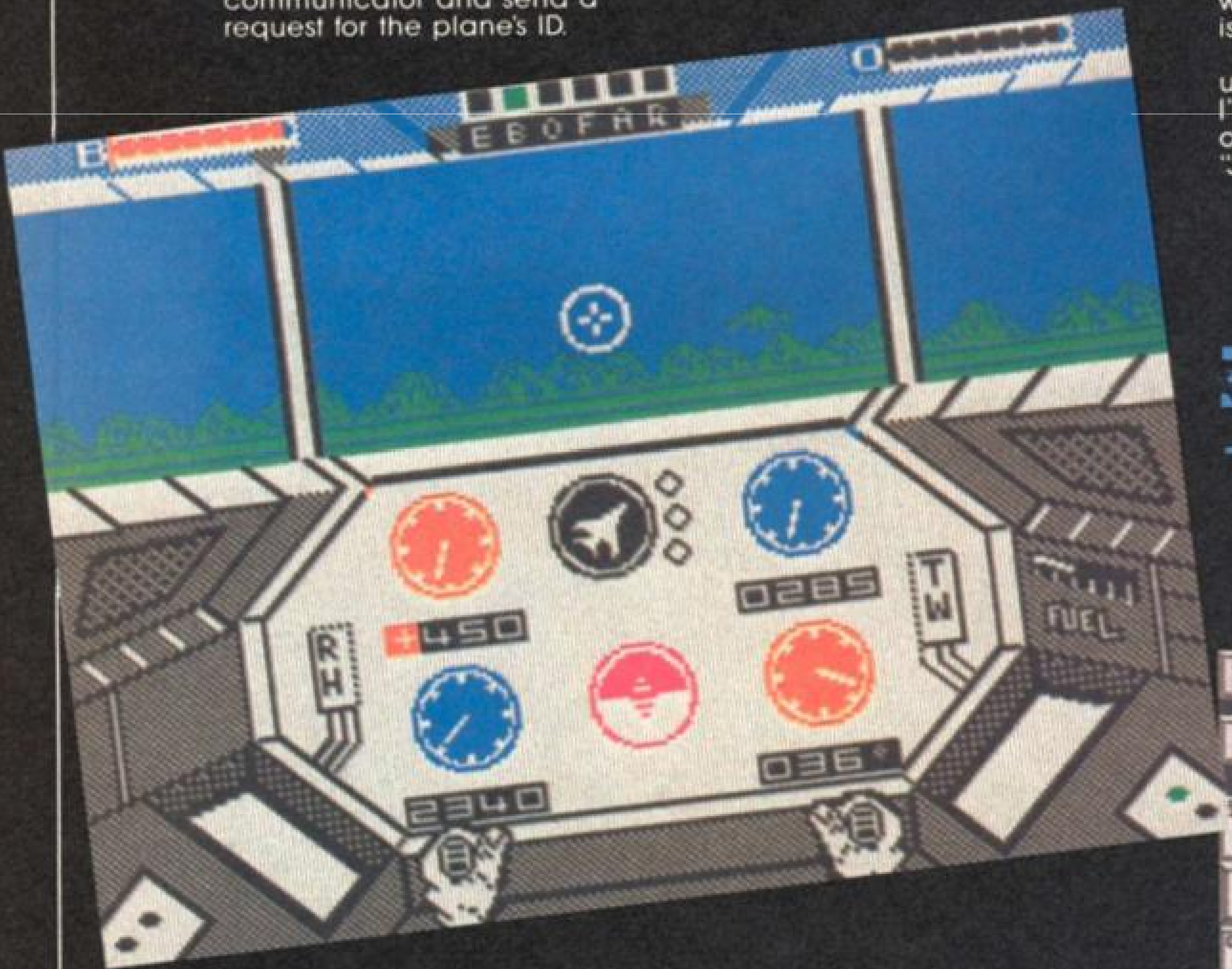
Get caught in the wrong place at the wrong time and it could be "Tough Luck Jimbo Baby".

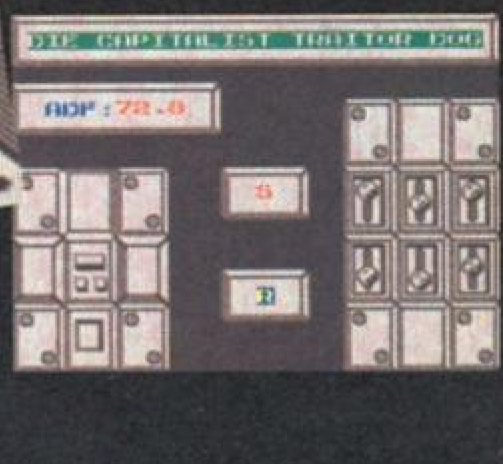
This Spectrum version is as good as the C64 original and is actually harder to play especially during the flight game since not only is the Gizmo harder to control but you will meet a lot more planes.

The game takes up both sides of the tape with a short delay while the base part of the game is loaded.

Infiltrator is a tough but undeniably addictive game and I'm sure we've not heard the last of the ridiculously overqualified "Jimbo Baby" McGibbets.

MONSTER HIT





QL COLUMN

Brian Becket reviews Digital Precisions Professional Astrologer.

Our fate dear Freddy lies not in our stars but in ourselves! This clever paraphrase of Shakespeare (whom I'm sure will forgive me) is for Digital Precision's ever-inventive and personable managing director Freddy Vachha who has released a new and highly sophisticated QL astrology program upon a weary and troubled world increasingly desperate to gain a measure of insight into whatever passes for destiny in these cynical days. **Professional Astrologer** sells for £59.95 or £69.95 depending upon an optional Professional Astronomer module. It comes on four microdrives or a single disc — I do not believe in astrology and rate its "scientific" worth as more or less equivalent to the old Roman custom of forecasting events by reading the entrails of sheep. While astrology is a good deal less messy and far more likely to keep the RSPCA off your back, its no less naive and — in today's world — an inexcusable exercise in the surrender of rationality to unreasoned belief. A great many believe in astrology to at least some extent (a great many people also once believed the earth to be flat) and there is clearly a market for a good program catering to their needs and Professional Astrologer is an indisputably good program. Besides, according to the (comprehensive) manual, neither Freddy Vachha nor chief program developer Elmar Duenssar believe in astrology either and I have no hesitation in recommending the quality of the package despite my total impressive piece of professional software and would seem to contain just about every feature required for the complex calculations required by those who take astrology seriously enough to buy a program to aid their work.

There's even plenty of scope for those of us who think it's all nonsense. Personally I intend to get some of my own back on all those glassy-eyed trendies who invariably appear at any party anywhere to stand around asking each other and the innocent alike what their signs are. "I don't know, I leave all that sort of thing to my computer", ought to be a good conversation stopper.

Professional Astrologer also allows for some phenomenally accurate modelling of the solar system and projections of planetary positions so that it is possible to use it for truly serious purposes (an astronomy package of equal power would and is one of the most impressive pieces of micro software that I have ever seen.

It is an excellent program/package which does the QL proud and has no real competitor on any micro. The amount of data that's been included in both the program and in the manual is impressive to say the least. The program's design allows for individual tailoring and is so flexible that highly personalised results are the order of the day. There's an editor and a printer driver facility (you can also save to Quill) that will produce length and detailed documents/reports of a professional standard.

All together, the package accounts for some 400K but is designed to run easily on an unexpanded QL and its versatility and power makes one appreciate the scope for high level computing that is actually available to even the most humble of home micro owners when the machine is treated as a serious piece of modern electronics and not just a portable games arcade. I wish some other software houses had taken such a professional approach to QL programming.

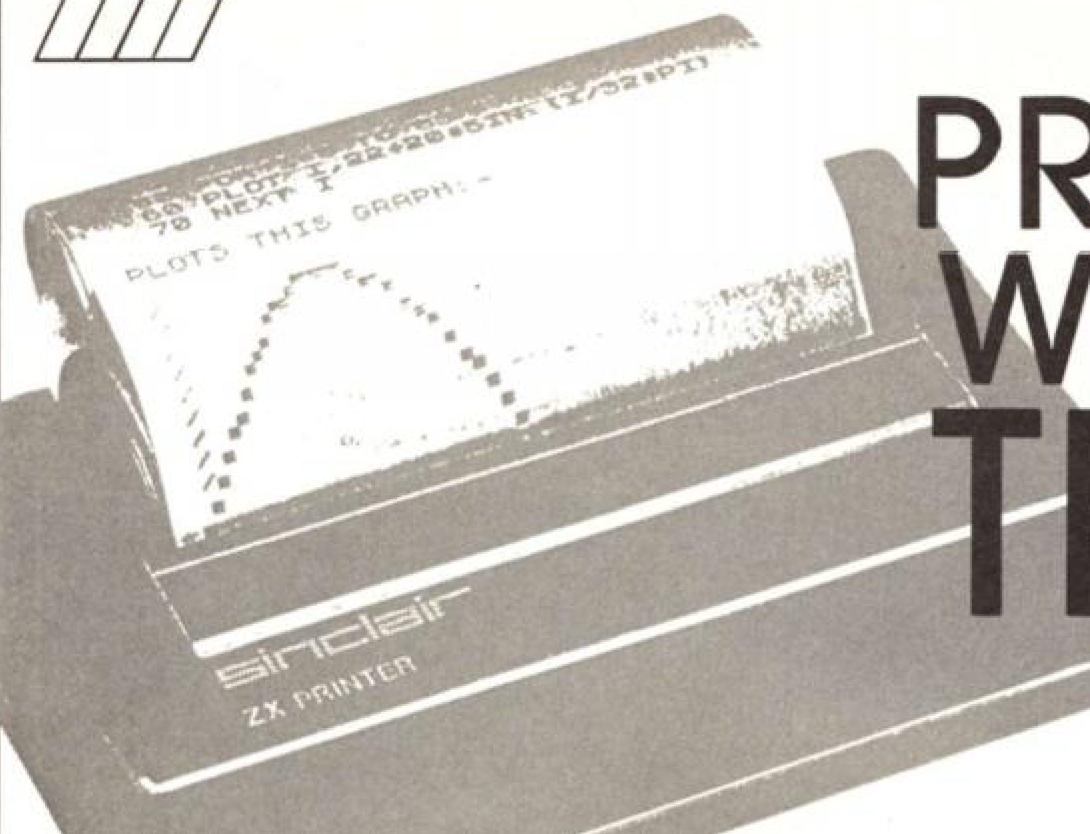
But — as one might have gathered from the opening line disbelief in the purposes for which it was designed.

Micro-astrology

The development of Professional Astrologer drew upon the consultancy services of honest-to-god astrologers so — if you're a true believer — have no fear of the scepticism voiced by the specialists who put it together or of the agnoticism of some reviewers who recommend it. It's fast, detailed and otherwise efficient but I can't suspend my disbelief enough to honestly comment on the results. My spies report very favourable if not enthusiastic attitudes towards Professional Astrologer in the star-gazer circle so, if you want a computer package for better gazing, this is undoubtedly the one to get. The package will do all the things a good astrologer should do save send you a bill. These include character evaluation from planetary positions at birth, character comparisons (check out your partner options), day-to-day and year-by-year forecasts. If you're intending to go into business as the neighbourhood Merlin, the word "professional" in the title is fully justified. The package is an be an absolute delight).

If we can look for more quality software for the QL as we know it in the days to come, Digital is one of the few companies who will provide it as most of the others are now seeking new claims in more promising gold fields. I got a couple of arcade games **(Droidzone** and **Blocklands)** shipped to me along with Professional Astrologer as well as a pile of other recent Digital products, so you're likely to be reading a fair amount about the company in coming columns.

For the moment, all I can say is that my 9-year-old finds Droidzone fast, exciting and fun, but that sadly Blocklands came on a duff microdrive that steadfastly keeps putting "bad or changed medium" on the screen. Just think where the QL might have been today, if Sinclair hadn't opted for those wretched microdrives. Dare I suggest that if Professional Astrologer had been available earlier, Sir Clive could have caught a glimpse of things to come and gone for a disc drive instead, or should we just take the old Greek view that while man cannot change his fate, he does have an inalienable right to bitch about it.



PRINTING WITHOUT TEARS

Thinking of buying a printer? Alan Davis maps out his chosen route through the hardware jungle.

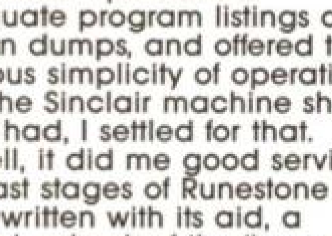
You've probably noticed — indeed, you could hardly help noticing — that the long-suffering Ray Elder spends more time in "Crosswires" unravelling problems with printers than he spends on any other single topic. Month after month, it seems, one poor soul after another sends up distress signals simply because some particular combination of hardware won't do what it was *bought* to do — print hard copy!

To these unfortunate people I have but one thing to say: I deeply sympathise! Speaking purely personally, I tend to view any add-on bit of hardware with initial scepticism and resent any time I have to spend getting it working as it should. Like most people, I buy a piece of equipment because I expect it to make my programming life *easier* — and days spent fiddling about just to make it work are, to me, wasted days.

To those who are in the midst of life-or-death struggles with a Mindboggler Mk2 printer and Brainbaffler Mk4 Interface, I can offer no comfort except to point you towards Ray Elder the Crosswires Wizard. However, to those who are still merely *thinking* of investing in a decent printer, but haven't yet committed themselves, I have a little story to tell which may be helpful.

Like many a Spectrum programmer, I started out in the early days with (no sniggering, please) a ZX printer. After the first one died, then a second, and finally a third, I decided to abandon this head-banging exercise in favour of something that actually worked. At that time the Seikosha GP50 had just become available, at what then seemed a reasonable price, and since it produced adequate program listings and screen dumps, and offered the glorious simplicity of operation that the Sinclair machine should have had, I settled for that.

Well, it did me good service. The last stages of Runestone were written with its aid, a sizeable chunk of the "sequel" (still far from finished), and a fair number of programs and articles for ZXC. The only snag was that its 32 character-per-line



output coupled with slightly ragged print quality made it pretty useless for word-processing, and I knew that sooner or later I was going to be forced into the purchase of a full-size job. And of course eventually the inevitable happened: the noble beast died noisily in mid-print, worn out by two years of continual use.

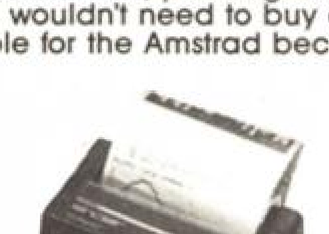
It was decision time. Should I go for a simple, cheap and straightforward replacement — or should I enter the jungle of full-size machines and interfaces which has been such a blight to so many? Reaching for the chequebook and the prayer mat, I chose the latter course.

Now I don't normally keep abreast of current hardware developments (perhaps inevitable in someone whose prime love is for elves and dragons), though I had a notion that the Centronics GLP was generally considered a reasonable machine at around £150. What I was hoping, you see, was that I could utilise the RS 232 on my Interface 1 to drive it. So I consulted a friend who keeps his finger firmly on the pulse of the hardware world — knowing that I'd get good advice (as I had on numerous previous occasions). What I said, effectively, was this:

"Given an Interface 1, which I already have, and the fact that I don't want to spend days writing utility programs to drive it, what printer should I buy? Would the GLP do?"

His answer startled me. Forget your Interface 1, he said. Go for the Kempston Centronics Interface "E" (which won't interfere at all with your microdrives) and use it to drive an Amstrad DMP 2000. That little lot will set you back £190 or so, but you'll get a superb NLQ option, a wide range of print styles, full Epson compatibility, normal or enlarged screen dumps at the press of a single key, tractor feed or single sheet loading, and I guarantee you'll be able to take it home and start printing within an hour or two.

Lacking faith, I demurred a bit at first. After all, the GLP was £40 cheaper, I said. But my adviser was adamant, pointing out that (a) I wouldn't need to buy a cable for the Amstrad because





Kempston provide one with the interface — which makes the difference only £30; (b) Getting screen dumps on the GLP/Interface 1 combination wouldn't be straightforward; (c) The print quality of the Amstrad knocked the GLP into a cocked hat — an opinion I could readily agree with after seeing a sample printout; and (d) He *couldn't* guarantee immediate hassle-free printing with the GLP/Interface 1.

I gave in, scraped together the extra cash (which hurt!) and took the plunge. But I didn't regret it. To give you some idea of just how easy it all was to set up, I need say no more than this: that within a few hours of arriving home with the stuff, I had printed out the entire text of an article for ZXC using Tasword 2, a lengthy assembler listing from Hisoft's DEVPAC, a BASIC program listing, and three or four enlarged screen dumps of excellent quality. The Amstrad printer manual is wonderfully detailed and clear, and the Kempston interface looks after you like a fairy godmother: just a handful of commands, simply and clearly explained in the instruction sheet, are all you need.

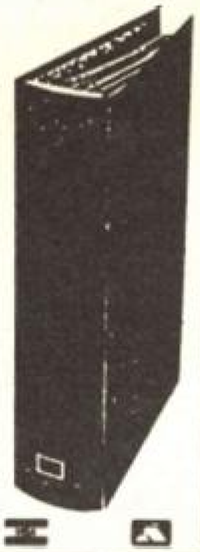
I can envisage two extreme reactions to what I'm saying here. The hardware freaks among you, who have no difficulty in connecting up 5 different printers with 20 different interfaces before breakfast, will wonder what on earth all the fuss is about. Of such people I can only stand in admiring awe, and apologise for wasting their time. At the other extreme, however, I can foresee a different reaction from those who might regard themselves as beginners, perhaps thus: "Yes well, it's all very fine for him. But he has a lot of programming experience, and what he finds obvious may drive me to distraction." To which I can safely reply — don't be





misled. I count as a complete novice in this field, and had no hands-on experience of printers other than the "ZX" type before this. If it was painless for me, then so will it be for you.

There are probably countless combinations of devices which operate just as easily as this one. I wouldn't know about those. But if you're envisaging spending up to £200 on a printer/interface combination; if you want a print quality which will enable you to produce more than merely acceptable word-processed material; if you want listings and screen dumps at the press of a key; and if you want to sleep easy, knowing that all this can be achieved with the minimum of fuss — well, then, I'm sure you could do a good deal worse than to go for the Kempston "E"/Amstrad DMP 2000 combination. If someone out there is helped by this information, then I rest content; and so, probably, will hard-worked Ray Elder.



STREAMS AND CHANNELS

Toni Baker sheds light on windows for all Spectrums.

In the last part we discussed what streams and channels were and what we could do with them. Also we introduced the concept of "user defined channels" — that is — new channels created in machine code to do various tasks not possible in any other way. Last month's episode included a program to CLOSE a user defined channel. Apologies are due here as unfortunately it was riddled with bugs, so I've listed a new version in this episode. You might like to play "Spot the bugs" by comparing last month's listing to this months — see how many you can find! The included version works, however, as does the rest of the program.

Windows

This article consists of a new user defined channel called a WINDOW. The concept of a window is very simple. Users of the QL will already be familiar with them. A window is a rectangular region of the screen which may be treated as if it were a whole screen — if you print text to a window it will only appear within the confines of this rectangle, and once the window gets completely full it will scroll independently of the rest of the screen. It is also possible to clear a window, in any colour scheme, without clearing the rest of the screen.

The program given will in fact cater for two different types of window, which I have called "Fast" and "Slow" windows (although in practice there seems to be no noticeable difference in speed between the two). A "Fast" window will do exactly what is described above, and no more. The standard character set is used (although this may be altered by changing the system variable (CHARS) at address 5C36) and all characters are eight pixels wide.

A "Slow" window, however, has two important extras. The first is left-justification. Put simply this means that words are treated as whole chunks of characters and will not be split up — thus if a word is too long to fit at the end of one line then it will be printed as a whole at the start of the next line. This is achieved by cleverly storing all of the characters in a buffer until the end of the word is reached, and then deciding whether or not it will fit on the line. This means that when you use a Slow window channel you never have to worry about spacing the words out to fit on the line — the channel does that all by itself.

The second improvement is that you are not restricted to eight bit wide characters — you can use seven bit wide; six bit wide; four bit wide even if you like. This means that you can fit more characters in each line than would normally be possible.

When a window becomes filled then it will scroll. Normally it will pause at this point, but it is possible to open a window which has the scroll pause disabled — in such a case the window will scroll automatically each time without pausing to wait for you. While a window is paused the screen will appear to freeze. You may then press either SPACE (which continues with the scroll) or BREAK (which will break out giving report message D BREAK - CONT repeats). Even if the scroll pause is disabled it will still be possible to break out at the point of scrolling by pressing BREAK.

The Spectrum, unlike the QL, does not come equipped with the statement CLS N (which clears a window), so I have had to make special provision to ensure that a window may be cleared easily from BASIC. I have used CHR$ 0 as a clear-window control, thus to clear a window attached to stream four it would merely be necessary to use the command.

PRINT 4;CHR$ 0;

and the job will be done. You can put other print items on the same line of course. All of the normal controls are allowed, including AT and TAB, all of the colour controls (PAPER, INK, FLASH and BRIGHT) as well as INVERSE. OVER is not incorporated, however, when used with SLOW windows OVER 1 will switch to printing with double-height characters, while OVER 0 will revert to printing with single-height characters.

As was stated in the last article, all of our new user defined channels will be identified by the fact that IX+05 will contain the two-byte value 1234h, where IX points to the channel information block. Figure one lists the channel information used by a window channel. Note that if the window is a Fast window then W_SCROLLS (IX+15) will be the last variable — IX+16 and beyond will not exist.

UDC's

The program commences with a few routines suitable for all user defined channels, not just windows. At address B000 the routine CLOSE_NEW will close the new channel which is attached to stream A. At address B061 is a routine which will close all user defined channels, but any data stored in buffers will be lost if this routine is used. At address B06D the routine OPEN_NEW will open any new user defined channel and attach it to a stream. To use this sub-routine the registers must be pre-assigned with the required values, as specified in the notes above the routine.

Then there are a couple of routines which will work with most user defined channels, though not necessarily all of them. CHR_TYPE (address B0B7) will expand keywords, and will count incoming control parameters. It will also set or reset the "Leading Space" bit (see Figure One) according to whether or not the character is a space. If, on return from this sub-routine the sign flag is set, it means that there is no more work to be done for this character. CHR_TYPE_2 (address B12F) will also deal with the comma control and the TAB function, and in a similar manner will also return with the sign flag set if work on the character is finished.

And then we come to the program itself. The program will work both on 16K/48K Spectrums, and on the Spectrum 128, in either mode — but when the program is run on a Spectrum 128 in 128K mode then it will be possible to define a window either on screen zero (the normal screen) or screen one. Note that the Spectrum 128 has no built-in software which may print onto screen one, so this program will make up for one of the few flaws in the new machine. Users of the Spectrum 128 must ensure that the machine stack resides below address BFFF otherwise the program will crash.

To open a window onto screen one it is only necessary to set bit 7 of the B register (they co-ordinate of the window) when opening the channel.

Here's how you open a window channel. Firstly you must assign the registers as defined in the notes above OPEN_WINDOW (address B212), and then all you have to do is call the sub-routine OPEN_WINDOW itself. Voila, the job will be done, and you can use the channel in BASIC just as easy (if not easier) as you can in machine code.

Figure 1

IX+00	W_OUT	Address of WINDOW output routine (=B4F1).
IX+02	W_IN	Address of WINDOW input routine (=15C4, REPORT_J).
IX+04	W_CHNAME	Name of channel (="W").
IX+05	W_IDEN	New channel identifier (=1234).
IX+07	W_CLOSE	Address of empty buffer routine (=B4D6 for Slow windows or 0052 for Fast windows).
IX+09	W_CHLEN	Length of channel information block.
IX+0B	W_FLAGS	Various flags, defined as follows:
	Bit 7:	Reset if leading space required, set otherwise.
	Bit 6:	Not used.
	Bit 5:	Set if scroll pause enabled, reset otherwise.
	Bit 4:	Set for Slow window, reset for fast window.
	Bit 3:	Set for INVERSE 1; reset for INVERSE 0;
	Bit 2:	Set if using double height; reset otherwise.
	Bits 1,0:	Number of control parameters required.
IX+0C	W_XCOORD	Current x coordinate of print position.
IX+0D	W_WIDTH	Number of characters per line.
IX+0E	W_YCOORD	Current y coordinate of print position.
IX+0F	W_HEIGHT	Height of window, in squares.
IX+10	W_PRPOS	Address within screen of current print position.
IX+12	W_HOME	Address within screen of top left hand corner of window.
IX+14	W_ATTR	Colours currently being used for window.
IX+15	W_SCROLLS	Counts number of scrolls allowed before scroll pause, +1.
IX+16	W_PIX	Position within square of current print position.
IX+17	W_CH_WID	Width of characters, in pixels.
IX+18	W_CHARS	Address of character set (0-7F) minus 100h.
IX+1A	W_UDG	Address of character set (80-A4).
IX+1C	W_WIDTH_8	Width of window, in squares.
IX+1D	W_LEN	Number of characters stored in buffer.
IX+1E	W_BUFFER	The buffer itself.

Listing 1

```
To close a new channel. On entry the A register must contain the stream number to be
closed. If called from the label CLOSE_CLEAR then the carry flag must be set if data in
buffers is to be sent, or reset if such data is to be lost.

                           ORG B000
37         CLOSE_NEW       SCF                     Signal "Data in buffer to be sent".
F5         CLOSE_CLEAR     PUSH AF                 Stack stream number.
08                         EX AF,AF'               F': stores the carry flag.
F1                         POP AF                  A:= stream number to close.
CD2117                     CALL 1721,STR_DATA_A    BC:= streams data for given stream;
                                                   HL: points to appropriate STRMS var.
E5                         PUSH HL                 Stack pointer to STRMS variable.
21EBFF                     LD HL,FFEB              HL:= -15h.
09                         ADD HL,BC
E1                         POP HL                  HL: points to STRMS variable.
D0                         RET NC                  Return with channels "K", "S", "R"
                                                   and "P", and also with streams which
                                                   are already closed.
DD2A4F5C                   LD IX,(CHANS)           IX: points to channel info area.
DD09                       ADD IX,BC
DD2B                       DEC IX                  IX: points to channel information
                                                   block for the given channel.
DD7E05                     LD A,(IX+05)
FE34                       CP 34
C0                         RET NZ
DD7E06                     LD A,(IX+06)
FE12                       CP 12
C0                         RET NZ                  Return if this is not one of our new
                                                   user defined channels.
3600                       LD (HL),00
23                         INC HL
3600                       LD (HL),00              Reset the STRMS variable, thus
                                                   closing the stream.
C5                         PUSH BC                 Stack displacement into channel
                                                   information area.
DD6E07                     LD L,(IX+07)
DD6608                     LD H,(IX+08)            HL:= address of "Send data in buffer"
                                                   subroutine.
08                         EX AF,AF'               Retrieve carry flag.
DC2C16                     CALL C,162C,CALL_JUMP   Send data if required.
DDE5                       PUSH IX
E1                         POP HL                  HL: points to channel info block.
DD4E09                     LD C,(IX+09)
DD460A                     LD B,(IX+0A)            BC:= length of channel info block.
C5                         PUSH BC
CDE819                     CALL 19E8,RECLAIM_2     Reclaim the memory used by block.
C1                         POP BC
3E10                       LD A,10                 A:= number of streams to consider.
21165C                     LD HL,5C16,STRMS_00     HL: points to stream zero variable.
5E         CLOSE_LOOP      LD E,(HL)
23                         INC HL
56                         LD D,(HL)               DE:= stream variable for next stream.
E3                         EX (SP),HL              HL:= stream variable for stream
                                                   just closed.
A7                         AND A
ED52                       SBC HL,DE
19                         ADD HL,DE
300B                       JR NC,CLOSE_NEXT        Jump unless the channel information
                                                   block for this stream has moved.
EB                         EX DE,HL
A7                         AND A
ED42                       SBC HL,BC
EB                         EX DE,HL                DE:= updated stream variable for
                                                   this stream.
E3                         EX (SP),HL              HL: points to 2nd byte of STRMS var.
2B                         DEC HL                  HL: points to 1st byte of STRMS var.
73                         LD (HL),E
23                         INC HL
72                         LD (HL),D               Store new value of streams variable.
E3                         EX (SP),HL
E3         CLOSE_NEXT      EX (SP),HL              HL: points to 2nd byte of STRMS var.
23                         INC HL                  HL: points to next STRMS variable.
3D                         DEC A
20E6                       JR NZ,CLOSE_LOOP        Loop back to consider remaining
                                                   streams.
F1                         POP AF                  Balance the stack.
C9                         RET                     Return.

To clear all new channels. Any data held in buffers will be lost, and the memory used by
all of the new user defined channels will be reclaimed.

                           ORG B061
3E10       CLEAR_NEW       LD A,10                 A:= number of streams to consider.
3D         CLEAR_LOOP      DEC A                   A:= stream number of next stream.
```

STREAMS

```
F5                      PUSH AF
A7                      AND A                   Signal "Data in buffers to be lost".
CD01BD                  CALL BD01,CLOSE_CLEAR   Clear the channel associated with
                                                this stream.
F1                      POP AF
20F7                    JR NZ,CLEAR_LOOP        Loop back till all streams cleared.
C9                      RET                     Return.
```

To open a new channel. On entry A' must contain the designated stream number; A must contain the name of the channel (an ASCII character code); BC must contain the length of the required channel information block; DE must contain the input address; HL must contain the output address; and IX must contain the address of a subroutine to send any data stored in buffers (or 0052h if this does not apply).

```
                        ORG BD6D
C5          OPEN_NEW    PUSH BC                 Stack length of channel info block.
DDE5                    PUSH IX                 Stack close buffer address.
F5                      PUSH AF                 Stack name of channel.
D5                      PUSH DE                 Stack input address.
E5                      PUSH HL                 Stack output address.
C5                      PUSH BC                 Stack length of info block (again).
08                      EX AF,AF'               A:= stream number to attach.
CD2117                  CALL 1721,STR_DATA_A    BC:= streams data for given stream.
78                      LD A,B
B1                      OR C
2802                    JR Z,OPEN_NEW_2         Jump unless stream already open.
CF17                    RST 08/DEFB 17          "O Invalid stream" error report.
E5          OPEN_NEW_2  PUSH HL                 Stack pointer to stream variable.
2A535C                  LD HL,(PROG)
2B                      DEC HL                  HL: points to 80h byte at end of
                                                channel information area.
CD5516                  CALL 1655,MAKE_ROOM     Make room for new channel info.
23                      INC HL                  HL: points to new channel info block.
22515C                  LD (CURCHL),HL          Make this the current channel.
E5                      PUSH HL
DDE1                    POP IX                  IX: points to channel info block.
23                      INC HL                  HL: points to second byte of info.
ED4B4F5C                LD BC,(CHANS)           BC: points to start of CHANS area.
A7                      AND A
ED42                    SBC HL,BC               HL:= required stream value.
EB                      EX DE,HL                DE:= required stream value.
E1                      POP HL                  HL: points to stream variable.
73                      LD (HL),E
23                      INC HL
72                      LD (HL),D               Assign stream variable with required
                                                value, thus opening the stream.
DDE5                    PUSH IX
E1                      POP HL                  HL: points to channel info block.
D1                      POP DE                  DE:= output address.
C1                      POP BC                  BC:= input address.
CDAEBD                  CALL BDAE,OPEN_STORE    Store these addresses in info block.
F1                      POP AF                  A:= name of channel.
77                      LD (HL),A               Store name of channel.
23                      INC HL
3634                    LD (HL),34
23                      INC HL
3612                    LD (HL),12              Signal "User defined channel".
23                      INC HL
D1                      POP DE                  DE:= close buffer address.
C1                      POP BC                  BC:= length of channel info block.
73          OPEN_STORE  LD (HL),E
23                      INC HL
72                      LD (HL),D               Store output addr or close addr.
23                      INC HL
71                      LD (HL),C
23                      INC HL
70                      LD (HL),B               Store input addr or block length.
23                      INC HL
C9                      RET                     Return.
```

To deal with parameters of control codes, and keyword tokens. This subroutine may be used with any user defined channel provided that (IX+0B) contains flags. Bits 1 and 0 count incoming parameters for use with colour controls, AT and TAB, while bit 7 is set if a space has just been printed to the channel. On return the sign flag will be set if no more needs to be done, otherwise the zero flag will be set if the character is a graphics character, and the carry flag will be set for block graphics and control codes.

```
                        ORG BOF7
4F          CHR_TYPE    LD C,A                  C:= character to "print".
DD7E0B                  LD A,(IX+0B)            A: contains various flags.
57                      LD D,A                  D: contains various flags.
E603                    AND 03                  A:= number of parameters expected.
2816                    JR Z,CTYP_GO            Jump if no parameters expected.
15                      DEC D
7A                      LD A,D
DD770B                  LD (IX+0B),A            Decrement parameter count.
E603                    AND 03
2805                    JR Z,CTYP_LAST          Jump if this is the last parameter.
FD7105                  LD (TVDATA)hi,C         Store this (middle) parameter.
185E                    JR CTYP_DONE            Jump to exit.
3A0E5C      CTYP_LAST   LD A,(TVDATA)lo         A:= original control code.
FD4605                  LD B,(TVDATA)hi         B:= middle parameter (if one exists).
1847                    JR CTYP_CTR_X           Jump to exit.
213B5C      CTYP_GO     LD HL,5C3B,FLAGS        HL: points to system flags.
1EA3                    LD E,"SPECTRUM"         E:= the first Spectrum 128 keyword.
CB66                    BIT 4,(HL)
2002                    JR NZ,CTYP_MODE         Jump if in 128K mode.
1EA5                    LD E,"RND"              E:= the 1st 16K/48K Spectrum keyword.
79          CTYP_MODE   LD A,C                  A:= character to print.
BB                      CP E
380F                    JR C,CTYP_NOT_TOK       Jump unless this is a keyword.
DD7E0B                  LD A,(IX+0B)            A: contains various flags.
07                      RLCA                    Bit 0:= leading space bit.
AE                      XOR (HL)
E601                    AND 01
AE                      XOR (HL)
77                      LD (HL),A               Assign leading space bit as reqd.
79                      LD A,C                  A:= character to print.
CD520B                  CALL 0B52,PO_TAUIG      Expand this keyword.
1838                    JR CTYP_DONE            Jump to exit.
FE80        CTYP_NOT_TOK CP 80
380D                    JR C,CTYP_NOT_GR        Jump unless this is a graphics chr.
DDCB0BBE                RES 7,(IX+0B)           Signal "Leading space will be reqd".
FE90                    CP 90
3003                    JR NC,CTYP_UDG          Jump if this is a user defined grphc.
BF                      CP A                    Set the zero flag.
37                      SCF                     Set the carry flag.
C9                      RET                     Return.
BF          CTYP_UDG    CP A                    Set zero flag/Reset carry flag.
C9                      RET                     Return.
FE20        CTYP_NOT_GR CP 20
380C                    JR C,CTYP_CTRL          Jump with control characters.
DDCB0BBE                RES 7,(IX+0B)           Signal "Leading space will be reqd"
2004                    JR NZ,CTYP_SPACE        Jump unless chr is SPACE.
DDCB0BFE                SET 7,(IX+0B)           Signal "Leading space not reqd".
A7          CTYP_SPACE  AND A                   Reset zero flag/Reset carry flag.
C9                      RET                     Return.
FE10        CTYP_CTRL   CP 10
3804                    JR C,CTYP_CTR_X         Jump to exit with codes 00 to 0F.
FE18                    CP 18
3803                    JR C,CTYP_PARAMS        Jump with codes 10 to 18.
FEFF        CTYP_CTR_X  CP FF                   Reset zero flag/Set carry flag.
C9                      RET                     Return.
320E5C      CTYP_PARAMS LD (TVDATA)lo,A         Store control code.
14                      INC D                   Signal "One parameter required".
FE16                    CP 16
3801                    JR C,CTYP_SET           Jump with codes 10 to 16.
14                      INC D                   Signal "Two parameters required".
DD720B      CTYP_SET    LD (IX+0B),D            Store number of reqd parameters.
F6FF        CTYP_DONE   OR FF                   Set the sign flag.
C9                      RET                     Return.
```

To deal with the comma control, and the TAB function as well. This subroutine may be used with any user defined channel, provided that (IX+0B) contains flags as above; that (IX+0C) contains the x coordinate (or column number) of the current print position; and that (IX+0D) contains the width of the line (or total number of columns allowed). Flags on return are as above, except that the sign flag will be set if comma or TAB have been dealt with.

```
                        ORG B12F
CDF7B0      CHR_TYPE_2  CALL B0F7,CHR_TYPE      Deal with ctrl params and keywords.
DD5E0D                  LD E,(IX+0D)            E:= width of line.
F5                      PUSH AF                 Stack the flags.
FE06                    CP 06
2806                    JR Z,CTYP_COMMA         Jump with comma control.
FE17                    CP 17
2810                    JR Z,CTYP_TAB           Jump with TAB control.
F1                      POP AF                  Restore the flags.
C9                      RET                     Return.
DD7E0C      CTYP_COMMA  LD A,(IX+0C)            A:= column number of print position.
3C                      INC A
E6F8                    AND F8
C608                    ADD A,08                A:= col num of next field.
BB                      CP E
3811                    JR C,CTYP_SPACES        Jump if column number in range.
7B                      LD A,E
180E                    JR CTYP_SPACES          Else tab past end of current line.
61          CTYP_TAB    LD H,C
68                      LD L,B                  HL:= TAB parameter.
1600                    LD D,00                 DE:= width of line.
ED52        CTYP_LOOP   SBC HL,DE
30FC                    JR NC,CTYP_LOOP
19                      ADD HL,DE               Reduce modulo line length.
7D                      LD A,L                  A:= column number to TAB to.
A7                      AND A
2001                    JR NZ,CTYP_SPACES       Jump unless column zero is required.
7B                      LD A,E                  Tab past end of current line.
```

AND

```
DDBE0C      CTYP_SPACES CP (IX+0C)
2807                    JR Z,CTYP_EXIT          Jump if required column reached.
F5                      PUSH AF
CD390C                  CALL 0C39,PO_SPACE      Print a space.
F1                      POP AF
18F4                    JR CTYP_SPACES          Jump back to see if finished.
F1          CTYP_EXIT   POP AF                  A:= control just dealt with.
F6FF                    OR FF                   Set the sign flag.
C9                      RET                     Return.
```

The following two entry points PAGE_0 and PAGE_7 will page in RAM page zero and RAM page seven respectively. The routines will have no effect unless called from a Spectrum 128 in 128K mode.

```
                        ORG B16C
C5          PAGE_0      PUSH BC
0600                    LD B,00                 B:= required page number.
1805                    JR PAGE_B               Jump forward.
C5          PAGE_7      PUSH BC
0607                    LD B,07                 B:= required page number.
F5                      PUSH AF
FDCB0166                BIT 4,(FLAGS)
280E                    JR Z,PAGE_EXIT          Jump if not in 128K mode.
3A5C5B                  LD A,(BANK_M)           A:= current page flags.
E6F8                    AND F8
B0                      OR B
01FD7F                  LD BC,7FFD              BC:= port no reqd to change page.
325C5B                  LD (BANK_M),A           Signal required RAM page.
ED79                    OUT (C),A               Actually change page.
F1          PAGE_EXIT   POP AF
C1                      POP BC
C9                      RET                     Return
```

The following three subroutines require that initially HL points to a byte on the screen, and will each adjust the value of HL in a different way. The first will point HL to the pixel immediately below the given one; the second will point HL down one line (ie down eight pixels); and the third will find the address of the corresponding attribute byte.

```
                        ORG B18C
24          DOWN_1      INC H                   Assume within a character square.
7C                      LD A,H
E607                    AND 07
C0                      RET NZ                  Return if this is so.
7D                      LD A,L
C620                    ADD A,20
6F                      LD L,A                  Assume crossing screen thirds.
```

```
D8                      RET C                   Return if this is so.
7C                      LD A,H
D608                    SUB 08
67                      LD H,A                  Adjust for within screen third.
C9                      RET                     Return.

                        ORG B19B
7D        DOWN_8        LD A,L
C620                    ADD A,20
6F                      LD L,A                  Assume within screen third.
D0                      RET NC                  Return if this is so.
7C                      LD A,H
C608                    ADD A,08
67                      LD H,A                  Adjust for crossing screen third.
C9                      RET                     Return.

                        ORG B1A5
7C        ATTR_ADDR     LD A,H
1F                      RRA
1F                      RRA
E606                    AND 06                  Isolate screen third number.
F6B0                    OR B0
CB14                    RL H                    Carry flag:= screen number.
1F                      RRA
67                      LD H,A                  HL:= attribute byte address.
C9                      RET                     Return.
```

The remainder of the programs in this article are concerned exclusively with WINDOW channels. The following subroutine will move the print position to the start of the (A+1)th line of the current window.

```
                        ORG B1B1
DDBE0F    LINE_A        CP (IX+W_HEIGHT)
D29F1E                  JP NC,1E9F,REPORT_B     Give error if line number too big.
DD360C00                LD (IX+W_XCOORD),00     Reset x coordinate.
DD770E                  LD (IX+W_YCOORD),A      Assign new y coordinate.
DD6E12                  LD L,(IX+W_HOME)lo
DD6613                  LD H,(IX+W_HOME)hi      HL:= address of top left hand corner.
A7                      AND A
2806                    JR Z,LINE_FOUND         Jump if line zero required.
47                      LD B,A                  B:= line number required.
CD9BB1    LINE_LOOP     CALL B19B,DOWN_8        HL:= address of next line.
10FB                    DJNZ LINE_LOOP          HL:= address of required line.
CD9AB3    LINE_FOUND    CALL B39A,STORE_ADDR    Store this address as print position.
DDCB0BFE                SET 7,(IX+W_FLAGS)      Signal "Leading space not required".
DDCB0B66                BIT 4,(IX+W_FLAGS)
C8                      RET Z                   Return if this is a "Fast" channel.
DD361600                LD (IX+W_PIX),00        Reset print position within chr sqr.
C9                      RET                     Return.
```

The next subroutine will clear one line of a window, given that A contains the attribute byte with which to clear; that HL contains the address of the line within the screen; and that BC contains the width of the line in squares, less one.

```
                        ORG B1DE
C5        CLW_LINE      PUSH BC                 Stack length of line, less one.
E5                      PUSH HL                 Stack address of line.
F5                      PUSH AF                 Stack attribute byte.
CDA5B1                  CALL B1A5,ATTR_ADDR     HL:= address of attribute line.
F1                      POP AF                  A:= attribute byte.
54                      LD D,H
5D                      LD E,L
13                      INC DE                  DE: pts to second attribute byte.
77                      LD (HL),A               Store first attribute byte.
EDB0                    LDIR                    Store remaining attribute bytes.
E1                      POP HL                  HL:= address of line.
C1                      POP BC                  BC:= length of line, less one.
3E08                    LD A,08                 A:= number of rows per line.
54                      LD D,H
5D                      LD E,L
13                      INC DE                  DE: points to second byte in line.
E5        CLWL_LOOP     PUSH HL                 Stack address of 1st byte in row.
D5                      PUSH DE                 Stack address of 2nd byte in row.
C5                      PUSH BC                 Stack length of line, less one.
3600                    LD (HL),00              Reset first byte.
EDB0                    LDIR                    Reset remaining bytes.
C1                      POP BC                  BC:= length of line less one.
D1                      POP DE                  DE:= address of 2nd byte in row.
E1                      POP HL                  HL:= address of 1st byte in row.
F5                      PUSH AF                 Stack loop counter.
CD8CB1                  CALL B18C,DOWN_1        HL:= addr of 1st byte in next row.
F1                      POP AF                  A:= loop counter.
14                      INC D                   DE:= addr of 2nd byte in next row.
3D                      DEC A
20ED                    JR NZ,CLWL_LOOP         Repeat for all eight rows.
C9                      RET                     Return.
```

This very short subroutine will fetch the width of the current window, measured in squares, into the C register.

```
                        ORG B206
DD4E0D    GET_WIDTH     LD C,(IX+W_WIDTH)       C:= width of window, in characters.
DDCB0B66                BIT 4,(IX+W_FLAGS)
C8                      RET Z                   Return with "Fast" windows.
DD4E1C                  LD C,(IX+W_WIDTH_8)     C:= width of window, in squares.
C9                      RET
```

To open a WINDOW channel and attach it to a stream. This subroutine will open either a "Fast" or a "Slow" window channel. The registers must be assigned on entry as follows:

For both types of channel:

A' The stream number to which the channel is to be attached.
A   The attribute byte used initially by the window.
B   The number of lines between the top of the window and the top of the screen.
C   The number of squares between the left of the window and the left of the screen.
D   The height of the window, in squares.
E   The width of the window, in squares.
H   00 if the scroll pause is disabled; FF if the scroll pause is enabled.
L   00 for a "Fast" window; FF for a "Slow" window.

For SLOW windows only:

BC' Address of normal character set, minus 100h.
DE' Address of graphics character set (beginning at CHR$ 80h).
H' The width of these characters, in pixels.

```
                        ORG B212
F5        OPEN_WINDOW   PUSH AF                 Stack attribute byte.
C5                      PUSH BC                 Stack coordinates of window.
D5                      PUSH DE                 Stack size of window.
E5                      PUSH HL                 Stack flags.
011600                  LD BC,0016              BC:= length of "Fast" chan info block.
DD215200                LD IX,0052              Signal "No buffer", presuming
                                                "Fast" channel.
2C                      INC L
2021                    JR NZ,OPWN_CREATE       Jump with "Fast" channels.
1600                    LD D,00                 DE:= window width, in squares.
62                      LD H,D
6B                      LD L,E                  HL:= window width, in squares.
29                      ADD HL,HL
29                      ADD HL,HL
29                      ADD HL,HL               HL:= window width in pixels.
D9                      EXX
7C                      LD A,H                  A:= character width in pixels.
D9                      EXX
5F                      LD E,A                  DE:= character width in pixels.
3EFF                    LD A,FF                 A:= -1.
3C        OPWN_LOOP     INC A
ED52                    SBC HL,DE
30FB                    JR NC,OPWN_LOOP         A:= width of window, in characters.
D9                      EXX
6F                      LD L,A
D9                      EXX                     L':= width of window, in characters.
5F                      LD E,A                  DE:= width of window, in characters.
211E00                  LD HL,001E              HL:= length of "Slow" channel info
                                                block, excluding buffer.
19                      ADD HL,DE               HL:= total length of chan info block.
44                      LD B,H
4D                      LD C,L                  BC:= total length of chan info block.
DD21D6B4                LD IX,B4D6,EMPTY_2      IX: points to empty buffer subrt.
21F1B4    OPWN_CREATE   LD HL,B4F1,WINDOW       HL: points to output subroutine.
11C415                  LD DE,15C4,REPORT_J     DE: points to input error routine.
3E57                    LD A,"W"                A:= name of this channel.
CD6DB0                  CALL B06D,OPEN_NEW      Open the channel.
E1                      POP HL                  HL:= flags.
D1                      POP DE                  DE:= size of window.
C1                      POP BC                  BC:= coordinates of window.
F1                      POP AF                  A:= attribute byte for window.
DD7714                  LD (IX+W_ATTR),A        Store attribute byte.
AF                      XOR A                   A:= 00
CB1D                    RR L
1F                      RRA
CB1C                    RR H
1F                      RRA
1F                      RRA
1F                      RRA                     Construct flags byte.
DD770B                  LD (IX+W_FLAGS),A       Store these flags.
DD730D                  LD (IX+W_WIDTH),E       Store window width.
DD720F                  LD (IX+W_HEIGHT),D      Store window height.
78                      LD A,B                  A:= y coordinate of window.
E618                    AND 18
F640                    OR 40
DD7713                  LD (IX+W_HOME)hi,A      Store high part of address of top
                                                left hand corner of window.
78                      LD A,B                  A:= y coordinate of window.
0F                      RRCA
0F                      RRCA
0F                      RRCA
```

The following subroutine prints a newline - ie it moves the print position to the left hand edge of the next line down.

```
                        ORG B2BB
CDC3B2    ENTER         CALL B2C3,ENTER_1       Print a single newline once.
DDCB0B56                BIT 2,(IX+W_FLAGS)
C8                      RET Z                   Return if using single height.
DD7E0E    ENTER_1       LD A,(IX+W_YCOORD)      A:= current y coordinate.
3C                      INC A                   A:= new y coordinate.
DDBE0F                  CP (IX+W_HEIGHT)
C2B1B1                  JP NZ,B1B1,LINE_A       Jump if in range to move print pos.
DDCB0B6E                BIT 5,(IX+W_FLAGS)
2812                    JR Z,SCROLL             Jump if scroll pause disabled.
DD3515                  DEC (IX+W_SCROLLS)      Decrement scroll count.
200D                    JR NZ,SCROLL            Jump unless scroll pause required.
DD7E0F                  LD A,(IX+W_HEIGHT)
DD7715                  LD (IX+W_SCROLLS),A     Re-initialise scroll count.
3E7F      SCROLL_PAUSE  LD A,7F
DBFE                    IN A,(FE)               Scan part of the keyboard.
1F                      RRA
38F9                    JR C,SCROLL_PAUSE       Pause until SPACE pressed.


E6E0                    AND E0
B1                      OR C
DD7712                  LD (IX+W_HOME)lo,A      Store low part of address of top
                                                left hand corner of window.
DDCB0B66                BIT 4,(IX+W_FLAGS)
2816                    JR Z,CLS_WINDOW         Jump with "Fast" windows.
DD731C                  LD (IX+W_WIDTH_8),E     Store window width, in squares.
D9                      EXX
DD750D                  LD (IX+W_WIDTH),L       Store window width, in characters.
DD7417                  LD (IX+W_CH_WID),H      Store character width, in pixels.
DD7318                  LD (IX+W_CHARS)lo,E
DD7219                  LD (IX+W_CHARS)hi,D     Store address of character set -100h.
DD711A                  LD (IX+W_UDG)lo,C
DD701B                  LD (IX+W_UDG)hi,B       Store addr of graphics chr set.
```

Control does not return immediately from the above routine, but continues into the next subroutine, whose purpose it is to clear a window and move the print position to the top left hand corner. It is the WINDOW equivalent to CLS.

```
                        ORG B292
AF        CLS_WINDOW    XOR A                   A:= 00.
CDB1B1                  CALL B1B1,LINE_A        Move print pos to top left corner.
CD06B2                  CALL B206,GET_WIDTH     C:= width of window, in squares.
2804                    JR Z,CLSW_CONT          Jump with "Fast" windows.
DD361D00                LD (IX+W_LEN),00        Clear buffer, abandoning contents.
DD460F    CLSW_CONT     LD B,(IX+W_HEIGHT)      B:= height of window, in squares.
DD7E14                  LD A,(IX+W_ATTR)        A:= attribute byte.
DD361501                LD (IX+W_SCROLLS),01    Reset scroll count.
CD71B1                  CALL B171,PAGE_7        Use RAM page 7 in case scr 1 in use.
0D                      DEC C                   C:= width of window, less one.
C5        CLSW_LOOP     PUSH BC
0600                    LD B,00                 BC:= width of window, less one.
F5                      PUSH AF
CDDEB1                  CALL B1DE,CLW_LINE      Clear next line of window.
F1                      POP AF
C1                      POP BC
10F5                    DJNZ CLSW_LOOP          Clear whole window.
C36CB1                  JP B16C,PAGE_0          Restore RAM page zero.
```

CHANNELS

The Windows program will continue next month.

THE DISCOVERY COLUMN



John Wase presents more useful routines for Opus Disc Drive owners.

Originally conceived as a serious computer, the Spectrum's outstanding success as a games machine has perhaps tended to obscure the very large number of people using it for serious purposes such as word processing. For the forgotten millions, here is an excellent utility, submitted by J. P. Riches of Warwick.

Like most Spectrum owners, I save my text to disc frequently. Thus I usually split a scientific paper into three or four files (numbered 1, 2, 3 and 4) with a further a, b, c, or d to identify the various versions. Indeed, this sort of system is implicit in the Tasword 3 manual (TEXT1, TEXT2, TEXT3). It is therefore a particular joy to have received a program which will erase all these files if you type in TEXT.

Directory

The program **(Figure 1)** first erases the file "directory" on disc 1 (line 115). A subtle touch in this line is the use of the original microdrive syntax (ERASE "m";1;"directory") rather than the shortened opus version (ERASE 1;"directory") as the former returns no error message if the file does not exist. A new file named "directory" is then opened and the disc catalogue is read into it. This ensures that the original catalogue file is not disturbed, and that the program can be used on twin-disc systems.

Main menu

Immediately after this, the main menu options are shown (lines 150-200), the principal command being to kill text files. In addition you can re-boot the disc (this gives the standard "run" program which one assumes has a program menu), CATalogue the disc if you've forgotten which files to kill, or, last, reset the machine.

Erase, erase . . .

The Textkill option then asks for a name of 10 letters or less, the number of letters selected being assigned to the variable "end". It then reads in each entry from the directory file in turn. The crunch line is 5140: this compares the strings (to "end") and if they agree, erases the appropriate file, continuing in a loop until the end of the directory is reached.

Twins

For users of twin discs, insert extra lines between 10 and 1000 to input a drive number and assign it to "d". Assuming that the program is on disc 1, insert "d" instead of the disc number in lines 122 5140.

And finally . . .

Don't forget to save the final, correct version with a star preceding the name; just prevents one losing the lot . . .

Opus Assembler conversion

The next suite of programs is from Ian Craig of Dundee. His listings will convert Hi-Soft's

Figure 1

```
   1 REM PROGRAM TEXTKILL
   2 REM J.P.RICHES, 1986
  10 CLEAR #
 100 PAPER 0: INK 7: BORDER 0: C
LS
 110 PRINT PAPER 1;AT 0,0;" OPUS
 DISCOVERY :  TEXT KILLER. "
 115 ERASE "m";1;"directory"
 121 OPEN #4;1;"directory"
 122 CAT #4;1
 123 CLOSE #4
 150 PRINT AT 8,3; PAPER 2;"MENU
"
 155 PRINT AT 11,3;"1. Kill text
 files."
 160 PRINT AT 13,3;"2. Re-boot d
isc."
 163 PRINT AT 15,3;"3. Directory
 disc."
 165 PRINT AT 17,3;"4. Return to
 basic."
 170 INPUT "COMMAND >"; LINE T$
 180 IF T$="1" THEN GO SUB 5000
 185 IF T$="2" THEN LOAD *"m";1;
"run"
 187 IF T$="3" THEN CLS : PRINT
PAPER 2;"DIRECTORY": CAT 1: GO S
UB 5510
 190 IF T$="4" THEN RANDOMIZE US
R 0
 200 RUN
5000 CLS
5005 PRINT AT 2,0; PAPER 2;"What
 characters"; PAPER 0;" do the t
ext": PRINT ,,"files begin with
?"
5006 PRINT ,,,,"You may enter be
tween 1 and 10": PRINT ,,"charac
ters - or enter ""MENU"" to"
5007 PRINT ,,"return to the menu
 page."
5008 INPUT "CODE >"; LINE A$
5009 IF LEN A$=0 OR LEN A$>10 TH
EN GO TO 5000
5010 IF A$="MENU" OR A$="menu" T
HEN RETURN
5020 CLS
5030 LET end=LEN A$
5105 LET total=0
5107 PRINT PAPER 1;"ERASURE REPO
RT"
5108 PRINT
5110 OPEN #4;1;"directory"IN
5115 PRINT #4;
5120 IF USR 432=0 THEN CLOSE #4:
 GO TO 5500
5130 INPUT #4;n$
5135 IF n$="" THEN GO TO 5150
5136 IF LEN n$<(end) THEN GO TO
5150
5140 IF n$(1 TO end)=A$(1 TO end
) THEN PRINT n$: LET total=total
+1: ERASE "m";1;n$
5150 GO TO 5115
5500 PRINT ,,,,total;" files era
sed."
5510 PRINT AT 21,0; PAPER 2;" Pl
ease press enter to continue "
5520 IF INKEY$="" THEN GO TO 552
0
5530 RETURN
9996 REM BACKUP PROGRAM TO DISC.
9997 STOP
9998 SAVE *"m";1;"TEXTKILL" LINE
 1
9999 RUN
```

GENS3 to run on a disc-based system. Unfortunately, for fairly obvious reasons, I couldn't test this one, as I have not got GENS3; nor, for that matter, a GP100A printer at home where this column is put together, but I thought it was so obviously useful that I have included it.

BASIC saver and loader

In the BASIC listing **(Figure 2)** line 5 allows a restart once you have dropped into BASIC (see line 100). Line 60 enters GENS and assigns the value of f to the result to indicate if a LOAD, SAVE or ERROR was requested. The auto-start line is line 10.

Assembly listing

The main explanations are included in the listing **(Figure 3)**. Just one or two small points remain. In the first place, the code looks disjointed, but this is partly due to having to fit it into the same space as the

Figure 2: BASIC saver and loader for GEN 3 Assembler conversion

```
  5 CLEAR : INPUT "RESTART ADDR? ";A: LET F=USR A: GO TO 65
 10 PRINT AT 5,0;"GENS3 OPUS DISC VERSION  HI-SOFT1985/86"
 20 INPUT "Load at what address ";a
 30 CLEAR a-1
 50 LOAD *1;"gens3d"CODE (PEEK 23730+256*PEEK 23731+1)
 60 LET f=USR (PEEK 23730+256*PEEK 23731+1)
 65 IF f=1 THEN GO TO 200
 70 IF f=2 THEN GO TO 300
 80 IF f=9 THEN GO TO 400
100 IF f>20 THEN STOP
200 REM PUT
210 SAVE *1;b$(2 TO )CODE a,b
215 CLEAR
220 LET f=USR (PEEK 23670+256*PEEK 23671)
230 GO TO 65
300 REM GET
305 PRINT '"Using:-";b$(3 TO )
308 IF b$(3 TO 5)="CAT" THEN CLS : CAT 1: PRINT £0;"Any Key to Return": PAUSE 0
 LET F=USR ((PEEK 23670+256*PEEK 23671)-1857): GO TO 65
310 OPEN £4;1;b$(2 TO )
320 PRINT £4;: LET l=USR 432-7
325 IF L>65535 THEN CLOSE £4: GO TO 400
330 POKE 23729,INT (l/256): POKE 23728,l-256*INT (l/256)
340 CLOSE £4
350 LOAD *1;b$(2 TO )CODE a
355 CLEAR
360 LET f=USR (PEEK 23670+256*PEEK 23671)
370 GO TO 65
400 REM ERROR
410 PRINT "Error or something!!"
420 LET f=USR (PEEK 23670+256*PEEK 23671)
430 GO TO 65
```

Figure 3

```
  10 ;ROUTINE TO REPLACE
  20 ;MDRV CALLS IN GENS3
  30 ;SET FOR GENS LOADED
  40 ;AT 50000
  50 ;THIS IS PUT FILE
  60 ;SAVED AS A CODE BLOCK
  70 ;STARTING AT A LGTH B
  80 ;B$ HOLDS DRIVE NO AND
  90 ;FILE NAME
 100 ;INDEX TABLE AT APPROX
 110 ;£E600 WHEN MONS AT
 120 ;50000 HAS TO BE ALTERED
 130
 140 ERRSP  EQU  £5C3A
 150 VARS   EQU  £5C4B
 160 RECLM  EQU  £19E8
 170 DIFF   EQU  £19DD
 180 MROOM  EQU  £1655
 190 PRT2   EQU  £CACE
 200 ELINE  EQU  23641
 210 SEED   EQU  23670
 220
 230
 240        ORG  52187
 250
 260
 270
 280 PUT    PUSH DE
 290        PUSH HL
 795        CALL NAMEB$-50000;*****
 800        LD   A,£62
 810        POP  DE
 820        CALL INSRT-50000;*****
 822
 824
 826
 828
 830 CONT   LD   A,£61
 840        POP  DE
 850        CALL INSRT-50000;*****
 860
 870        LD   BC,61;WILL HOLD GENS START ONCE INIT.****
 872        EXX
 874        LD   BC,1
 876        EXX
 878 EXIT1  LD   HL,SEED
 880        LD   (HL),C
 890        INC  HL
 900        LD   (HL),B
 910 EXIT   LD   SP,(391);****
 920        EXX
 930        POP  DE
 940
 950        PUSH BC
 960        EXX
 970        JP   362;****
1000
1010 ;SUBROUTINES NAME INTO B$
1012
1014        ORG  52309
1016
1020 NAMEB$ LD   HL,(VARS)
1025        LD   BC,13
1030        CALL MROOM
1040        INC  HL
1050        LD   A,£42
1060        LD   (HL),A
1070        INC  HL
1080        LD   BC,10
1090        LD   (HL),C
1100        INC  HL
1110        LD   (HL),B
1120        INC  HL
1130        LD   DE,7856;*****
1140        LD   B,10
1142 TEST   LD   A,(DE);TEST FOR ENDLINE
1144        CP   £0D
1146        JR   Z,SPACE
1148        LD   (HL),A
1150        INC  HL
1152        INC  DE
1154        DJNZ TEST
1156        RET
1158 SPACE  LD   A,32;SPACE
1160 SP1    LD   (HL),A
1162        INC  HL
1164        DJNZ SP1
1166        RET
1170
1310
1320
1330
1340
1350
1360
1370 ;THIS IS "GET" FILE
1380 ;FOR GENS AT 50000
1390 ;IT CONSITS OF 2 PARTS
1400 ;EXIT FROM PART 1
1410 ;WITH BC= LOAD/SAVE/ERROR
1420 ;VARIABLES ARE
1430 ;A=CODE START ,SEED=RET ADDRESS
1440 ;B$=NAME
1450
1460
1470
1480
1490
1500
1510
1520
1530        ORG  52653
1540
1550
1560 GET    NOP
1562        NOP
1564        NOP
1570        CALL NAMEB$-50000;****
1580        LD   DE,(7916);****
1590        LD   HL,(54);****
```

```
1600        AND  A
1610        SBC  HL,DE
1620        JP   NZ,REN-50000;****
1630 NOTREN LD   A,£61
1640        LD   DE,(7916);STRT ADDR*****
1650        CALL INSRT-50000;LET A=STRT ADDR
1652        LD   BC,RET1-50000;*****
1660 OUT    EXX
1670        LD   BC,2
1680        EXX
1690
1700        JP   EXIT1-50000;******
1710
1720
1730
1740
1750        ORG  52557
1760
1770
1780
1790 REN    LD   A,£61
1800        LD   DE,(54);FIN ADDR ****
1810        CALL INSRT-50000;A>STRT ADDR ****
1820        NOP
1822        NOP
1824        NOP
1830        LD   BC,RET2-50000;****
1840        JP   OUT-50000;*****
1842
1850
1860 RET1   CALL 381;STR RET TO BASIC;****
1870        LD   DE,(23728)
1880        LD   HL,(7916);****
1890        ADD  HL,DE
1900        NOP
1910        LD   (54),HL;ADJUST END TXT POINTER;****
1920        JP   £C5AD-50000;****
1930 ;A=CODE START,SEED=RSTART
1940
1942
1944        ORG  51918
1946
1950 RET2   CALL 381;STORE BASIC POINTER;****
1960        LD   HL,(54);OLD FINISH TXT;****
1970        LD   DE,(23728)
1980        AND  A
1990        ADC  HL,DE
2000        JP   C,ERREXT-50000;****
2010        NOP
2020        LD   (54),HL;****
2030        LD   HL,£E229-50000;GENS EDITOR WORKSPACE****
2040        LD   A,£4E;N FOR RE-NUMBER
2050        LD   (HL),A
2060        LD   B,4
2070        LD   A,1
2080        LD   C,0
2090 LOOP2  INC  HL
2100        LD   (HL),A
2110        INC  HL
2120        LD   (HL),C
2130        DJNZ LOOP2
2140        LD   IX,£E234-50000;SET FOR RE-NUMBER****
2150        LD   (IX+0),C
2160        JP   £C5C1-50000;****
2170
2180
2190
2200 ;INSERT A NUMERIC VAR
2210
2220
2230        ORG  52602
2240 ;DE TO HOLD NUMBER
2250 ;AND A TO HOLD VAR NAME
2260 INSRT  LD   HL,(VARS)
2270        LD   BC,6;LGTH OF NUM VAR
2275        PUSH DE
2280        CALL MROOM
2285        POP  DE
2290        INC  HL
2300        LD   (HL),A
2310        XOR  A
2320        INC  HL
2330        LD   (HL),A
2340        INC  HL
2350        LD   (HL),A
2360        INC  HL
2370        LD   (HL),E
2380        INC  HL
2390        LD   (HL),D
2400        INC  HL
2410        LD   (HL),A
2420        RET
2430
2440
2450 ;COME HERE ON ERROR
2460
2470 ERREXT EXX
2480        LD   BC,9
2490        EXX
2500        LD   BC,61;*****
2510        JP   EXIT1-50000;*****
2520
2530 ;ALTER VECTOR TABLE FOR
2540 ;NEW ADDRESSES
2550
2560
2570        ORG  £E5FD
2580
2590        DEFW £088E
2600        DEFW £0894
2610
2620        ORG  £E605
2630
2640        DEFW £089D
2650
2660        ORG  £E623
2670
2680        DEFW £089A
2690
2700        ORG  £E627
2710
2720        DEFW £08AC
2730        DEFW £08B3
2740
2750        ORG  £E641
2760
2770        DEFW £07A6
2780        DEFW £07AC
2781        DEFW £077F
2782        DEFW £0782
2783        DEFW £078C
2784        DEFW £0790
2785        DEFW £0793
2790
2800        ORG  £E675
2810
2820        DEFW £0A49
2830        DEFW £0A4C
2840        DEFW £0A6E
2850
2860        ORG  £E67D
2870
2880        DEFW £0A82
2890        DEFW £0A01
2900        DEFW £0A04
2910        DEFW £091B
2920        DEFW £0A0A
2930        DEFW £0A61
2940
2950        ORG  £E68B
2960
2970        DEFW £0A65
2980        DEFW £0A68
2990        DEFW £0A74
3000        DEFW £0A77
3010        DEFW £0A7A
3020
3021
3022
3023 ;THE FOLLOWING ARE
3024 ;ADDITTIONAL VECTORS
3025 ;WHICH WILL ADD TO THE
3026 ;LENGTH OF GENS 3
3027 ;NEW LENGTH IS 10046
3028
3030        ORG  £EA7F
3040
3190        DEFW £0A0D
3200        DEFW £0A10
3210        DEFW £0A17
3220        DEFW £0A1C
3230        DEFW £0A1F
3240
3250        DEFW £0000
```

microdrive routines. For this reason the NOP's, too, must remain.

Getting going

First, type in the BASIC listing. SAVE it to disc with an auto-start at line 10. Next, do CLEAR 25999 and load GENS code at 26000; reload the GENS code at 50000 as well. Now do a cold start at 26000 and enter the assembly listing **(Figure 3).** Once this is complete and checked, do an assembly to generate the Opus version of the code from 50000 onwards. Finally, SAVE this as "gens3d"CODE, 50000, 10046. This, then, completes the conversion.

Dump

And, for good measure, here is Ian's assembler listing of a COPY routine for the Seikosha GP100A printer **(Figure 4).** Printer dump routines operating through the Opus port are always welcome, but please note my comments at the bottom of the column.

Finally

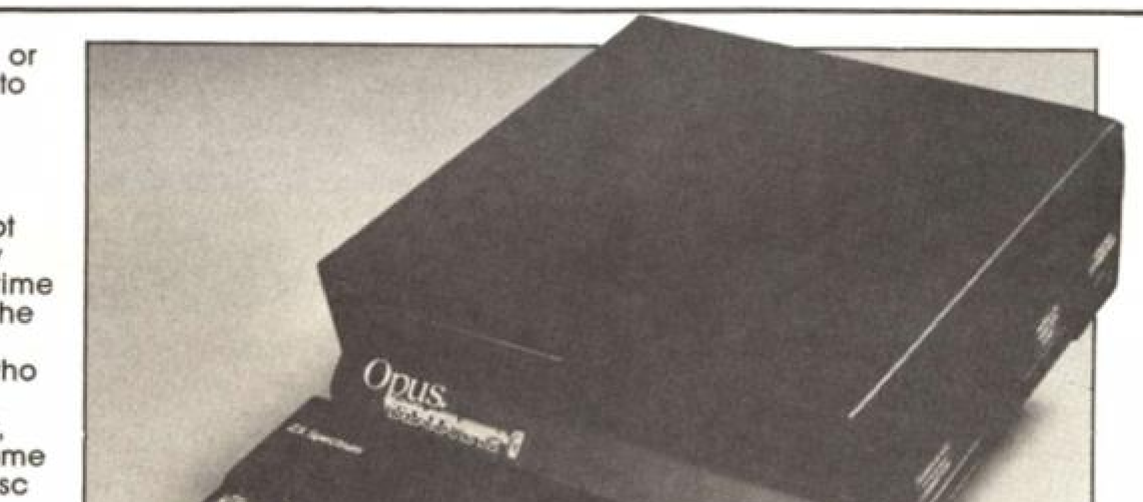
Just one gripe. If you send in an assembly listing, please, oh

Figure 4

```
 . ;COPY ROUTINE FOR A
 20 ;SHEISKOWA GP100A PRINTER
 30 ;WHEN USED WITH DISCOVERY
 40 ;1 DISC INTERFACE
 50 ;
 60 ; I.CRAIG JUN 1986
 70
 80 ;AN OPEN£ 3;"B" MUST
 90 ;HAVE BEEN DONE IN BASIC
100 ;TO SET UP THE CHANNELL
110
120
130        ORG  23296
140
150 START  LD   A,3;STRM 3
160        CALL £1727;GET STRM PARAM
170        LD   A,B
180        OR   C
190        CP   £10
200        JP   Z,£1725
210        LD   A,3
220        CALL £1601
230
240        CALL TAB  ;SPACE
250        LD   D,27;27*7=190
260
270        LD   BC,£C000    ;STRT AT 192,0 TOP LEFT
280 CONT   PUSH DE
290        PUSH BC
300        CALL SUBR;SEND BYTE
305        RST  £10
310        POP  BC
320        POP  DE
330        INC  C
340        JR   NZ,CONT;0=END OF LINE
350        LD   A,B;STEPS OF 7
360        SUB  7 ;LET B=B-7
370
380        LD   B,A
390
400
410
420        LD   A,£0A
430        RST  £10
440        LD   A,£0D;START NEXT LINE DOWN
450        RST  £10
455        CALL TAB
460        DEC  D
470        JR   NZ,CONT;IF NOT 25 PASSES COMPLETE
480
490 CONT2  LD   E,2      ;2BITS LEFT
500        LD   D,0;BTM BITS 0
510        PUSH DE
520        PUSH BC
530        CALL NEXT
531        RES  7,A
532        LD   B,5
534 ROT2   RRCA
536        DJNZ ROT2
537        SET  7,A
538        RST  £10
540        POP  BC
550        POP  DE
560        INC  C
570        JR   NZ,CONT2
580        LD   A,£0F;BACK TO ALPHA MODE
590        RST  £10
600        LD   A,£0D
610        RST  £10
620        RET  ;BACK TO BASIC
630
640
650
660 SUBR   LD   E,7;DOTS/LINE
670 NEXT   PUSH BC
675        LD   A,£C0;196 LIMIT
680        CALL £22AC;FIND BIT
690
700        LD   B,A;BIT NO.
710        INC  B;ROTATE TO BIT 0
720        LD   A,(HL);GET BYTE
730 ROT    RLCA
740        DJNZ ROT;ROTATE BIT INTO C FLAG
750        RR   A;ROT INTO CARRY FLAG
760        RR   D;ADD IT IN
770        POP  BC
780        DEC  B;NEXT COORD
790        DEC  E
800        JR   NZ,NEXT
810        SCF
820        RR   D
830        LD   A,D;COMPLETED BYTE TO A
840
850        RET
860 TAB    PUSH BC
870        LD   B,8
880        LD   A,£0F;ALPHA
890        RST  £10
900 TAB1   LD   A,32 ;SPACE
910        RST  £10
920        DJNZ TAB1
930        POP  BC
940        LD   A,8 ;GRAPHICS
950        RST  £10
960        RET
```

please do include either hex or preferably decimal numbers to be poked in, along with the necessary BASIC loading program. And do send full documentation. To be faced with several programs and not know what they do is a pretty daunting prospect and very time consuming, so full details of the programs are essential.

Thanks to all the readers who have sent in Opus programs. Especially welcome are short, snappy routines which use some unique facility of the Opus disc — I've not seen much with random access yet. So do please keep on sending your discs in. See you next month.



Erratum: Discovery Column Dec Page 1, Col 2, "disc names must start with #"

GET A PIECE OF THE ACTION!

Your chance to win Mikrogen's new shoot-em-up set in the mobster ridden streets of 1920's America.

Mikrogen, who recently celebrated five years in the games software business have just released Cop Out, an all action shoot-em-up that pits you against the Mafia in an operation to track down illegal stills during the prohibition era.

Chosen as the game for Mikrogen's National Computer Games Championship final held at the Savoy in London, Cop Out requires deadly accuracy and split second timing to uphold the law and rid the city of the Mafia.



There are 10 prizes of the game to be won and to help the first prize winners tot up their collossal law enforcing high scores they will each receive a Mikrogen pocket calculator as well.

Ten runners up will also receive a copy of the game.

Piece of the action

To get a piece of the Cop Out action all you have to do is answer three simple questions on the prohibition period.

1. A Speakeasy was
a) a loud hailer
b) a microphone
c) an illegal drinking den

2. Bootlegging was
a) a fashionable dance
b) making illegal hootch
c) embroidered insoles favoured by leading gangsters

3. Spats were
a) footwear favoured by leading gangsters
b) a slang word for bloodstains
c) ammunition

Send your entries to Cop Out Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.

The competition is open to all ZX readers except employees of Argus Specialist Publications, Chase Web and Mikrogen.

The editor's decision is final and no correspondence can be entered into. Please remember to write your answers on the back of the envelope. The closing date is February 10th.

Cop Out Competition

My answers are (A, B or C)

1. ____

2. ____

3. ____

Name ....

Address ....

....

....

Send your entry to Cop Out Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.

COMPETITION

the war of the shires

To battle! Alan Davis puts the last sliver of warlike code in place.

Although there's been much rumbling of thunder during the last couple of months, there's been little in the way of real action. The breaking of the storm is long overdue — so let's get straight down to business.

To get yourself a "finished" version of "War of the Shires" (finished, that is, as far as my own development of it is concerned) all you need to do is load in the BASIC program from last month's article and then add the program lines I've given here (Listing 1). There's no need to include the REM statements of course (since the program will run faster without them); their purpose is purely to help you to relate the new material to the stuff you have already. Notice that the new line 110 replaces the old one — which was in any case only a RETURN instruction. Once the typing is finished, save everything to tape just as you did last month when making your "temporary" version: SAVE "SHIRES" LINE 8000, with the map



the war of the shires

array, machine code, and udg bytes patched on at the end of the tape in the correct order.

If you take a look at Listing 1, you'll see that the main additions comprise the following:
(a) computer controlled movement of the enemy armies;
(b) a battle routine;
(c) an extension to the "explore location" routine;
(d) an extra batch of command options.

Apart from a few other lines interspersed through the program for general "tidying-up" and "housekeeping" purposes, that's all there is to it. In play, of course, you'll find that the Shires are now somewhat less peaceful than they used to be...

Carnage

To see the most immediately obvious difference, load in the game, select the "View Map" option for Roland, and press key "1" to enter ARMIES mode. You can then, if you wish, sit back and watch the invasion taking place as the forces of Darkness move out from the west. Because the game runs in real-time, the enemy armies move relentlessly even when you're doing nothing — and the map is continually updated as you watch, with news of battles flashed up on the screen as they occur. You can observe the carnage for as long as you like, unless the character you're controlling is attacked; if this occurs, the program jumps out of ARMIES mode to give you full details of the battle.

It's worth bearing in mind that the presence of an army symbol on the map can mean that either one army, or several, are currently at that location; this means that if a particular army moves between two already heavily occupied locations, you won't actually see any change on the map.

The other important difference concerns what happens when you instruct your current character to explore keeps or villages. Provided the leader of the corresponding Shire has been recruited to the cause, the local populace will assist with provisions and extra men-at-arms (unless you have an army which is already so large that it clearly needs no help...). A particular keep or village will render this service once, and only once, because their resources are limited — but each Shire leader can accumulate stocks of provisions and carry them with him for later distribution to the men as required.

Sample screens from the battle torn War of the Shires.

```
** Roland of Greenways **

Roland stands in a pine forest
in the Shire of Harland. He
commands 1248 men-at-arms who
are in excellent spirits. At
present they are in good fettle.
Roland is confronted by 1210
warriors of Darkness, preparing
to do battle.

OPTIONS

1: Change character
2: View map or Move
3: Explore the pine forest
4: Attack!
7: Distribute provisions

< Any other key for fresh news >
```

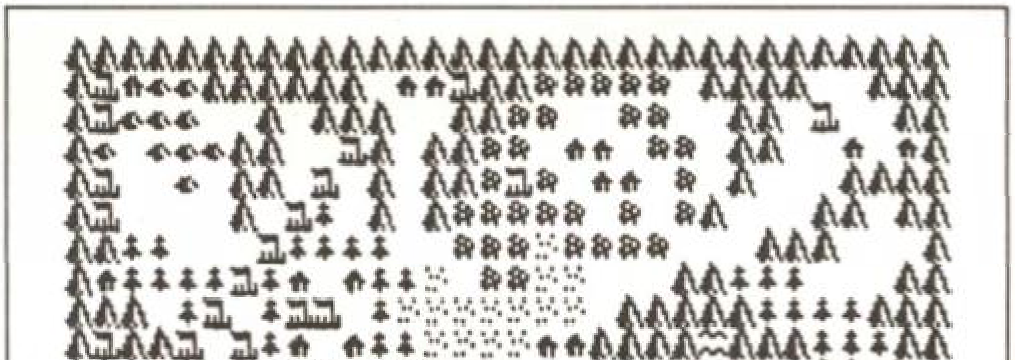
```
** The Battle of Harland **

The clash of steel upon steel
rings through the Shire of
Harland.

In the Battle of Harland, Roland
and his 1213 men fought bravely
against an army 1168 strong.

Roland slew 263 of the enemy,
and lost 86 men.

< ANY KEY TO CLEAR >
```



WARGAMES

Once play begins, the strategy is up to you. Obviously, Roland will need to round up as many warriors and provisions among the freemen of Greenways as he can. Before setting off to warn the recruits the leaders of other Shires. As play progresses you'll be able to change continuously between recruited characters, so as to conduct the war in whatever way you see fit.

As for the battles themselves — well, not surprisingly the outcome of a battle is more likely to be in your favour if you're defending a keep rather than fighting in the open. The battles occur only between individual armies; the tactics of the enemy are to wear down their quarry by attacking in waves, one army at a time — so that even if several enemy armies are present, you'll find your men engaged with them only one at a time. Each battle takes its toll on the participants, in terms of losses of course but also of stamina and possibly morale. A heavy loss will lower morale, which is bad news — though subsequent light losses will restore it progressively. Stamina can be recovered either by rest, or by nourishment. The game ends either by victory (the fall of the Keep of Darkness) or defeat (all Shire ledears dead). In case you hadn't realised: the Keep of Darkness is inaccessible for most of the game; you gain access to it via the "endgame" (more of which later) when the enemy has suffered catastrophic losses.

Customised warfare

As I promised last month, you now have a game which is readily playable in its own right, though with ample scope for you to develop it further. To that end, let's take a look at the material in Listing 1, starting with the additions to the "explore" routine. It's clear that we need the program to keep a record of those keeps and villages which have already supplied food and men, and an effective way to do this (though not the most economical in terms of memory) is to have a kind of dummy "copy" of the map. This takes the form of the array m$(22,32), in which each array element tells us something about the goings-on at a particular location.

When this array is dimensioned initially, all its elements will of course contain CHR$32. If the resources of a particular village or keep have been exhausted, however (or if the people are scattered as a result of the demise of their leader), we can arrange for the appropriate element of m$() to be changed to some other value to record that fact. I chose CHR$ 1 for this job — though one could use other values to store different kinds of information of course.

Getting down to the details, then, we find first of all a check in line 1100 to see if the current location —co-ordinates (x,y) — contains a village or keep. If it does, then line 1105 checks to see whether the leader of this shire is still alive and inserts CHR$ 1 at m$ (y,x) if he isn't — which, as we'll see, prevents recruitment there. Finally, line 1110 causes villagers to turn up with recruits and supplies provided that (a) the leader of this shire has already been recruited; (b) the current character's army isn't too big; (c) m$ (y,x) isn't CHR$ 1; and (d) a gang of recruits isn't already waiting here. Provided all these conditions are satisfied, the number of recruits (chosen at random between 100 and 255) is inserted into m$ (y,x) in the guise of the appropriate character code, and your character's stocks of provisions (stored in a(char,9)) increased by 1.

This little lot immediately produces extra options for your character of course; if you'd like to glance at the additions to the "character description and options" section, you'll see that lines 545 and 546 deal with these. And while we're on the subject, notice that if the option to recruit reinforcements is taken, then m$(y,x) is set at CHR$ 1 to prevent further recruitment (line 565).

The rest of the "explore" routines, lines 1200 to 1220, concern only the endgame in which a tunnel through the mountains to the Keep of Darkness becomes accessible. This simply involves setting the variable "tu", creating the entrance to the tunnel at the location (xtu,ytu) — which varies somewhat from one game to another — and finally permitting movement through the tunnel when the appropriate location(s) are explored. The existence of this tunnel, by the way, explains my suggestion in an earlier article that you shouldn't modify the map in this region. If you'd happened to plonk a mountain so that it coincided with the tunnel entrance, then it would make life a shade difficult . . .

The real-time movement of enemy armies is dealt with in lines 102-125. This routine is called repeatedly either immediately after one of your own commands, or in any case about once per second while the program awaits your desperate finger-prod at the keyboard (see the sub-routine at line 20 in last month's listing).

While reading what follows, remember that each enemy

Listing 1

```
  20 IF battle THEN RETURN
  45 IF battle THEN RETURN
  50 IF battle THEN RETURN
  99 REM ************************
         Real Time Action
        ************************
 102 LET m=FN r(65): IF NOT u(m,1) THEN RETURN
 103 LET fk=u(m,3): IF NOT a(fk,1) THEN LET u(m,3)=FN r(11): RETURN
 110 LET ux=u(m,4): LET uy=u(m,5): LET vx=ux+(ux<a(fk,4))-(ux>a(fk,4)): LET u(m,4)=vx: LET vy=uy+(uy<a(fk,5))-(uy>a(fk,5)): LET u(m,5)=vy
 115 LET t(uy,ux)=t(uy,ux)-u(m,1): LET t(vy,vx)=t(vy,vx)+u(m,1): IF armies THEN LET chr=CODE x$(uy,ux): PRINT INK 0;AT vy-1,vx-1;CHR$ 154;AT uy-1,ux-1; INK FN i(chr);x$(uy,ux) AND NOT t(uy,ux)
 120 IF vx<>a(fk,4) OR vy<>a(fk,5) THEN RETURN
 122 IF armies THEN LET z$=FN n$(fk)+" is engaged in battle": PRINT #1; FLASH 1;AT 1,0,,AT 1,0;TAB FN p();z$: PRINT BRIGHT 1;AT vy-1,vx-1; OVER 1; INK 8; FLASH 1;" ";CHR$ 8;: PAUSE 300: PRINT BRIGHT 0; OVER 1; INK 8; FLASH 0;" ": GO SUB 9
 125 LET ind=1: GO TO 2005-(5 AND fk=char)
 399 REM ************************
              Movement
        ************************
 400 IF NOT a(char,1) THEN RETURN
 402 IF battle THEN RETURN
 499 REM ************************
       Character Description
            and Options
        ************************
 501 IF NOT a(char,1) THEN LET z$=FN f$(char)+" was slain in the Battle of "+FN s$(FN c(y,x)): GO SUB 30: GO SUB 9: PAUSE 500: RETURN
 512 IF y=13 AND x=2 AND NOT t(y,x) THEN GO TO 6000
 515 IF t(y,x) THEN LET z$=FN n$(char)+" is confronted by "+STR$ t(y,x)+" warriors of Darkness, preparing to do battle": GO SUB 30
 530 IF t(y,x) THEN PRINT TAB 2;"4: Attack!"
 545 IF m$(y,x)>CHR$ 100 AND a(char,1)<2500 THEN PRINT TAB 2;"6: Recruit ";STR$ CODE m$(y,x);" men"
 546 IF a(char,9) THEN PRINT TAB 2;"7: Distribute provisions"
 563 IF i$="4" AND t(y,x) THEN LET ind=0: GO SUB 2000
 565 IF i$="6" AND a(char,1)<2500 AND m$(y,x)>CHR$ 100 THEN LET a(char,1)=a(char,1)+CODE m$(y,x): LET m$(y,x)=CHR$ 1
 570 IF i$="7" AND a(char,9) THEN LET a(char,2)=a(char,2)+50: LET a(char,9)=a(char,9)-1: IF a(char,2)>249 THEN LET a(char,2)=249
 999 REM ************************
          Search Location
        ************************
1100 IF x$(y,x)<>CHR$ 148 AND x$(y,x)<>CHR$ 149 THEN GO TO 1200
1105 IF NOT a(FN c(y,x),1) THEN LET m$(y,x)=CHR$ 1
1110 IF a(FN c(y,x),6) AND a(char,1)<2500 AND m$(y,x)<>CHR$ 1 AND m$(y,x)<CHR$ 100 THEN LET m$(y,x)=CHR$ (100+FN r(155)): LET z$="Presently, "+STR$ CODE m$(y,x)+" freemen of "+FN s$(FN c(y,x))+" arrive bearing provisions for the army, and offer themselves for recruitment": GO SUB 30: LET a(char,9)=a(char,9)+1
1200 IF NOT tu AND tot<15000 THEN LET z$="A mortally wounded warrior of Darkness staggers in and collapses. In his delirium, he speaks of the existence of a secret tunnel through the mountains to the Keep of Darkness": GO SUB 30: LET tu=1: LET xtu=7: LET ytu=10+FN r(6): GO SUB 9: PAUSE 0: GO SUB 10: RETURN
1210 LET in=1 AND x<5 AND y>11 AND y<15
```

```
1220 IF tu AND ((NOT in AND x=xt
u AND y=ytu) OR (in AND x=4 AND
y=13)) THEN LET a(char,4)=(4 AND
 NOT in)+(xtu AND in): LET a(cha
r,5)=(13 AND NOT in)+(ytu AND in
): LET z$=FN n$(char)+" discover
s a secret tunnel. After passing
 through many dark passages, "+(
"he" AND char<>8)+("she" AND cha
r=8)+" emerges into daylight": G
O SUB 30: LET in=NOT in
1999 REM ************************
                  Battle!
         ************************
2000 CLS : LET armies=0: LET bat
tle=1: LET fk=char: LET vx=a(fk,
4): LET vy=a(fk,5): LET z$="The
Battle of "+FN s$(FN o(vy,vx)):
GO SUB 15: LET z$="The clash of
steel upon steel rings through t
he Shire of "+FN s$(FN o(vy,vx))
: GO SUB 30: PRINT : IF ind THEN
 LET ind=0: GO TO 2003
2001 FOR i=1 TO 70: IF u(i,1) AN
D u(i,4)=vx AND u(i,5)=vy THEN L
ET m=i: GO TO 2003
2002 NEXT i
2003 LET z$="In the Battle of "+
FN s$(FN o(vy,vx))+", "+FN n$(fk
)+" and "+("his " AND fk<>8)+("h
er " AND fk=8)+STR$ a(fk,1)+" me
n fought bravely against an army
 "+STR$ u(m,1)+" strong"
2004 GO SUB 30: PRINT
2005 LET us=(INT (a(fk,1)*a(fk,2
)*a(fk,3)/62500)): LET them=INT
(u(m,1)*u(m,2)/300): LET us=us*(
1+(x$(vy,vx)=CHR$ 149 AND vy<>13
 AND vx<>2))
2010 LET sup=us-them: LET theirl
oss=FN r(100)+(sup AND sup>0): I
F theirloss>u(m,1) THEN LET thei
rloss=u(m,1)
2015 LET t(vy,vx)=t(vy,vx)-their
loss: LET u(m,1)=u(m,1)-theirlos
s: LET tot=tot-theirloss
2017 LET ourloss=FN r(100)-(sup
AND sup<0): IF ourloss>a(fk,1) T
HEN LET ourloss=a(fk,1)
2018 LET a(fk,1)=a(fk,1)-ourloss
2020 IF NOT a(fk,1) THEN LET x$(
a(fk,8),a(fk,7))=" "
2025 LET a(fk,3)=a(fk,3)-(50 AND
 ourloss>100)+(50 AND ourloss<50
): IF a(fk,3)<=0 THEN LET a(fk,3
)=0
2026 LET a(fk,2)=(a(fk,2)-50) AN
D a(fk,2)>=50
2027 LET u(m,2)=(u(m,2)-50) AND
u(m,2)>=50
2030 IF a(fk,3)>248 THEN LET a(f
k,3)=248
2035 IF fk<>char THEN RETURN
2040 IF NOT a(fk,1) THEN LET z$=
FN n$(fk)+" was slain": GO SUB 3
0: GO TO 2090
2050 LET z$=FN n$(fk)+" slew "+S
TR$ theirloss+" of the enemy, an
d lost "+(STR$ ourloss AND ourlo
ss)+("no" AND NOT ourloss)+(" me
n" AND ourloss>1)+(" man" AND ou
rloss=1): GO SUB 30
4999 REM ************************
            Select Character Menu
         ************************
5016 IF NOT lives THEN GO TO 650
0
5999 REM ************************
                  Victory!
         ************************
6000 LET z$="The Keep of Darknes
s is captured, and the foe defea
ted. Let all freemen of the Shir
es rejoice!": PRINT : GO SUB 30
6010 GO TO 6010
6499 REM ************************
                  Defeat!
         ************************
6500 LET z$="The Shires are defe
ated by the forces of Darkness.
All is lost": PRINT : GO SUB 30
6510 GO TO 6510
```

army is assigned a specific "target" character, whose code is stored at the appropriate row in the third column of the array u() — the only exceptions being armies 66 to 70 inclusive, who have the doubtful honour of standing guard at the Keep of Darkness.

This is what each line does:

102: Choose at random one of the 65 mobile armies ("m") provided it still survives!

103: Check this army' "target" (fk); if it's already been despatched to the Great Shire in the Sky, assign another target at random.

110: Store the armies current location in the temporary variables vx, uy. Compare these with the location co-ordinates of the target, and compute new co-ordinates vx, vy which will bring the army one location closer to its target. Move the army to this new location by changing u(m,4) and u(m,5).

115: The array t(22,32) stores the TOTAL number of enemy warriors at each and every location, and obviously the program needs to update this array as an army moves — which is what is happening here. In addition to this, if the program is currently in ARMIES mode, then the map graphics need updating too — which is what the remaining conditional instructions are for.

120: If the army hasn't reached its target, this turn is completed.

122 & 125: Let battle commence! A suitable message is flashed if the program is in ARMIES mode, before battle is actually joined. The variable "ind" is used to distinguish between attacks initiated by the enemy (ind=1) and those mounted by the player (ind=0).

Into battle

And this, naturally, brings us to what might be described as the core of the game — the battle sequence. The same routine is used whether the player or the enemy launches the attack — the only difference being the entry point; battles initiated by the enemy and taking place "behind the scenes" enter at line 2005, with the appropriate variables already assigned by the movement routine. When text print-out is necessary (i.e. when the currently controlled character is involved in the rough stuff), entry is at line 2000. The actual combat begins at line 2005, and follows a pretty simplistic system.

Two quantities "us" and "them" (!) are calculated for the two sides according to number, stamina, and morale (this last not applying to the enemy) of warriors, adding an appropriate boost to the value of "us" if the home team is defending a keep. Losses on both sides are calculated on the basis of the difference between "us" and "them" — lines 2010 and 2017 — with the appropriate checks to ensure that losses don't exceed the actual size of the army (negative numbers of men wouldn't exactly be conducive to realism!) Line 2015 updates the tally of enemy warriors at the location of the battle, as well as the variable "tot" which maintains an on-going record of the total number of enemy warriors in the entire game. The remaining lines deal with such concerns as the destruction of a keep if its owner is killed (line 2020); the adjustment of the morale of the home team according to losses sustained (lines 2025 and 2030); and the adjustment of the stamina or strength of both armies (lines 2026/7). All that remains is the printing of any text needed.

The rest of the program needs no explanation, I think, although a word about the "IF battle THEN RETURN" lines might be in order. These are necessary because we don't know precisely what the player might be doing in the event of a sudden attack by the enemy, and so the program could be jerked into the battle sequence from any of several sub-routines. The "IF battle . . ." instructions ensure that the program is channelled back to "Character description" mode at the end of a battle — whatever else it might have been in the middle of.

If you play the game exactly as given, then with a bit of thought and planning you could reasonably expect to achieve a victory within about a couple of hours — though you'll doubtless suffer some tragic losses, and things may seem desperately hair-raising in mid-game. If you find it too easy, then probably the simplest way of making it more challenging is to increase the number (or perhaps the initial strength) of the enemy armies. Personally, I don't worry too much about this side of things. I find that the fun of this kind of game (which incidentally is no less fun for the author to play than for anyone else) lies not so much in the winning as in the sense of involvement with the characters during play; a noble and desperate defence, ending in defeat, can be just as imaginatively satisfying as a glorious victory. This of course is one of the great strengths of games such as "Lords of Midnight", and my own debt to Mike Singleton's genius is pretty obvious!

Over to you, then. There's plenty of memory left for you to add ideas of your own. You might like to add some adventure-style puzzles, perhaps, to add depth to the plot. The endgame could profit from extra ideas, since it's no more than a sledgehammer job on the Keep of Darkness at present, and you might like to make use of the extra character (Ulric) whom I included in last month's listing but haven't, in fact, used.

But just a moment! What's that? The faint rumble of thunder? Roland is donning his armour! Deepmeads has fallen! To battle!

Now where *did* I put that sword . . .?

David Nowotnik puts the new version of the Quill to the test.

The Quill Adventure Writer
Gilsoft
£22.95

Mention the name 'Quill' to any QL owner, and he or she will automatically think you are talking about the QL's word processor. So, it is somewhat unfortunate that another product for the QL should have the same name. But the 'Quill Adventure Writer' has developed such a good reputation on the Spectrum that Software house Gilsoft must believe that this reputation will overcome any confusion on the QL.

'Quill' is a machine code program which provides users with a framework for producing their own text-only adventure programs. It is menu-driven and quite easy to use. Using a logical approach and a little practise anyone can produce an adventure game in a matter of a few hours.

The product is available in two price formats. At £22.95 Quill is supplied with a printed manual and the programs on one disc (3 ½" or 5 ¼") or two microdrives. Alternatively, customers can send £10.95 and supply the magnetic media. In this form, the manual is supplied on one of the cartridges (or disc) as an ASCII file, but there is three sheet introduction to get you started.

The review copy was supplied on two microdrive cartridges, with the manual on cartridge. This manual can be loaded into a text editor (e.g. from Metacomco), although you will need a RAM expansion to squeeze it all in. A hard copy can be obtained by COPYing the file to a serial port connected to a printer.

An electronic manual of this sort is a very poor substitute for a proper manual. I COPYed the manual file to an RX80 printer. It failed to paginate correctly, and margins came out all wrong. The end result was something just useable, but very untidy. And if you haven't got a printer, then you have nothing to refer to while learning to use the program. Still, Gilsoft do give you a choice of proper hard copy or the cheaper, but much nastier 'soft' copy.

The manual recommends using an EXEC command to start the adventure writer program; but there is a perfectly good BOOT program on the cartridge which does this for you.

First steps

In use, 'Quill' turned out to be quite impressive. The manual takes you step by step in writing a simple adventure with only six locations. And with that training, really complex adventures can be easily within the grasp of any adventure programmer.

The program depends upon an adventure 'database' being constructed. A basic database called 'start' is provided on the cartridge, and the first action in setting up any new adventure is load this in. The sequence of actions is then something like this:

First set the screen size (by adjusting the border), then permanent ink and paper colours. For each of the six locations enter a 'location text'. This is the message which appears when you enter any new location, for example:

"I am in the Hall. The Kitchen is to the West and the Dining Room is to the North."

These texts can be inserted in a single colour, or mixture of colours. Word wrapping is not provided, so users have to be careful to avoid word splitting. But text can be amended as well as inserted in the database, and there is good protection system, which prevents you accidentally inserting new text to location which already has some. During text entry, the cursor can be moved up and down, as well as right and left, for rapid editing. But the cursor had a tendency to disappear when moving it rapidly right to left.

Each location is assigned a location number; the next step in constructing the adventure is to link up locations, to give the 'Movement Table'. This is done through another selection from the main menu. It's quite logical. If from location number 5, you can move east to 6, west to 4, and north to 8 then you'd type in for location five:
E6W4N8

Movements up, down and to the diagonal points of the compass are also possible. You can also give names to locations. If location 8 is "HALL", then this name can be added to the movement table, so, when playing the game, you can say 'GO NORTH' or 'GO TO HALL', and the computer will respond to both.

With all the location texts and movement data entered, it is possible to carry out a test of the database. Again from the main menu, a single keypress takes you into your basic adventure game. The database has an elementary vocabulary (to which you can add many more words), so you can take yourself around the locations; the computer will respond with the appropriate location messages, and tell you if it fails to understand a command, or if it cannot move in any one direction. And all you have had to do to get this far in constructing your game is enter some simple text and movement data.

Diagnosis

In testing the game, the user is given a 'diagnostics' option. Supposedly, this provides extra information while running a test, but in the review version, I found that requesting 'diagnostics' made no difference to the information presented on the screen.

Entering objects, and locating them is also very simple. New words can be added to the basic vocabulary provided; synonyms are easily identified; they just have the same word number in the vocabulary file. The text recognition routines are rather basic. They can recognise only up to two words, and only the first four letters of a word are significant.

Probably the most difficult item to understand is 'the event table'. Again, this is another item from the main menu, and it sets conditions to the computer's response to the player's commands. In a way, it can be likened to 'Archive'; you use some high level, quite specific keywords to instruct the computer how to respond to commands. A large part of the manual is devoted to explaining its complexity, and users may need to read this section a few times to grasp all the elements of this programming 'language'. A second manual file, useful for reference purposes, is available on one of the cartridges, once you have grasped all the concepts of the first manual.

As the whole process of programming your adventure could take several hours, SAVEing and LOADing your database are available options to give you a well deserved rest!

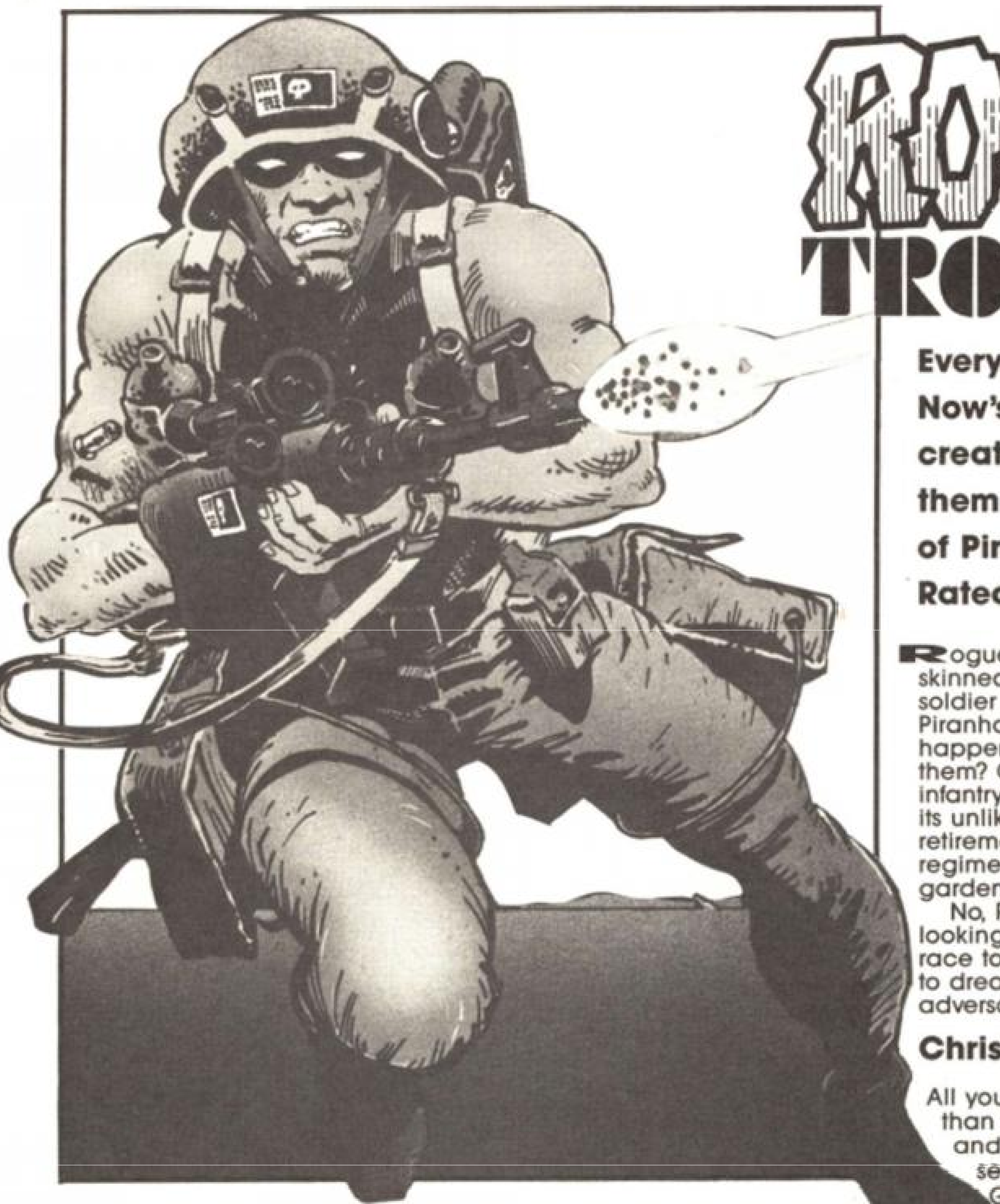
Gilsoft make no demand for royalty, so if you believe your adventure program is good enough for sale, you are free to do so. Of course, your adventure will be text only. In response to competition, Gilsoft have produced 'add-ons' for the Spectrum version of Quill, to allow graphics to be added.

Gilsoft are developing an 'Illustrator' upgrade for the QL version of Quill; whether it ever reaches the market will depend on the demand for Quill. Whatever the outcome of that development, budding adventure program writers could do well cutting their teeth on 'The Quill'.

Gilsoft, 2 Park Crescent, Barry, South Glamorgan CF6 8HD.

QL QUILL QL QUILL QL QUILL QL QUILL

ROGUE IN VOGUE



ROGUE TROOPER

Everyone likes a villain. Now's your chance to create a whole race of them and win a copy of Piranha's Monster Hit Rated Rogue Trooper

Rogue Trooper, the blue skinned genetically engineered soldier is up against the Norts in Piranha's new game. But what happens when he's vanquished them? Genetically engineered infantrymen being what they are its unlikely he will seek early retirement from his one man regiment and potter around the garden.

No, Rogue Trooper will be looking for another evil alien race to defeat and it's your job to dream up his next set of adversaries.

Christen the aliens

All you have to do is in less than 20 words, name and describe these seemingly invincible foes. Giving them a memorable name dripping with infamy is good for starters.

Then in 20 words describe how Rogue Trooper will deal with them. Perhaps he'll need a new kind of power of weapon to sort out the fiendish foes you have created.

There are 30 copies of Rogue Trooper to be won for ZX readers who can vividly describe their alien scourge and Rogue Troopers antidote.

Send your entries to Rogue Trooper Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB.

The Competition is open to all ZX readers except employees of Argus Specialist Publications, Chase Web and Piranha. The editor's decision is final and no correspondence can be entered into.

The closing date is February 10th, 1987.

Rogue Trooper Competition

1) I christen my Aliens: ....

2) They are really awsome because (in less than 20 words):

....

....

3) Rogue Trooper will be more than a match for them because:

....

....

Name: ....

Address: ....

....

....

Send your entry to Rogue Trooper Competition, ZX Computing Monthly, No 1 Golden Square, London W1R 3AB

Ray Elder with good tidings for 81 programmers.

Great news for all ZX81 programmers in the guise of Paul Kecskemety who has formed Lightning Software and who hopes to produce a range of tapes for our wee dinosaur.

His first product is for us programmers and is a program called ZXTENDED BASIC. Priced at a reasonable £2.50 inc P&P it consists of approx 5K of machine code additions to BASIC held in a line 0 REM.

In all there are 42 new commands which can be called from BASIC and some are very useful, they include invert, scrolls, variable lists, memory used, restore/read/data, renumber, delete and a host of string and number handling utilities.

As programming is the most frequent of our ZX81 activities, mainly due to there not being many games programs around, this is a must for serious users who don't want to delve into machine code but want more versatile commands.

Some of the syntax is a bit awkward to handle, but this is rather a minor complaint, instructions are on three pages of typed foolscap and are a bit on the brief side. Contact: Lightning Software, 95 Penton Drive, Cheshunt, Herts EN8 9RU.

Other programs including hi-res games are promised — watch this space . . .

REM lines

Steven McDonald of E. Lothian sent us a routine to create REM lines, this is a little more versatile than the one I printed some 9 months ago and is worth adding to your collection.

It works by building a program line in the work space area (E-LINE), and then using the ROM routines to enter it into the BASIC area — very neat.

Storing and saving

I thought we'd have a short discussion this month for the less expert among us, many of whom exist but we rarely think of them as we wander around our machine code world.

Storing text and numbers, especially those which remain constant is done in a very inefficient manner in the ZXG1. Each time a LET A$="ANYTEXT" instruction is used then the computer stores this twice, once in BASIC memory and again in the variables area of memory.

Numbers are worse! Each number not only takes the character space it occupies but also uses a byte value 126 to indicate a numeral and five bytes to store it in floating point form.

This short routine will demonstrate this:

```
10 LET X=25
20 FOR I=16514 TO 16535
30 PRINT PEEK I
40 NEXT I
```

When run there is a list of numbers, the first four numbers are the line number and line length, the next is the code of the keyword LET followed by the code for x,=,2 and 5.

After this comes the number indicator 126, the five floating point bytes and finally the number 118 which indicates the end of line.

This can be overcome if you are using integers by using VAL "25" to save two bytes, but if I am using integer constants then I either store them in a string by LET X$=CHR$ 27 etc. and retrieve them with VAL as required. Combining this with the string slicing of the ZX81 gives a versatile, efficient numerical store.

Advantage

However the ZX81 has one great advantage over more sophisticated machines, it saves all its variables with the program.

So to make the most of this I always have a routine to initialise variables which will remain constant throughout a program run, usually those which are for PRINT AT's etc. and add new ones to it as required. Then when the program is finished or when I start running short of memory, after running that area of the program (GOTO 9000 for example) I delete all these lines.

The disadvantage is that you MUST NEVER use CLEAR or RUN but start the program with GOTO 10 or whatever line you wish.

```
9980 FAST
9981 DIM A$(49)
9982 LET B$="2A1440360023360023
360023360023336EA23361C545D1301010
0EDB0367F23367623221A40221C40ED7
B04403B3BC31806"
9983 FOR A=1 TO 49
9984 LET A$(A)=CHR$(16*CODE B$+C
ODE B$(2)-476)
9985 LET B$=B$(3 TO )
9986 NEXT A
9987 PRINT "ENTER A LINE NUMBER
(4 DIGITS)"
9988 INPUT B$
9989 IF LEN B$<>4 THEN GOTO 9988
9990 LET A$(5)=B$(1)
9991 LET A$(8)=B$(2)
9992 LET A$(11)=B$(3)
9993 LET A$(14)=B$(4)
9994 PRINT "HOW MANY CHARACTERS
AFTER REM ?"
9995 INPUT A
9996 LET A$(25)=CHR$(A-256*INT(A
/256))
9997 LET A$(26)=CHR$(INT(A/256))
9998 SLOW
9999 RAND USR (PEEK 16400+256*PE
EK 16401+6)
```

Listing — to create REM lines.

To make this less likely I save programs so they auto start by including two lines such as

```
9998 SAVE "PROGNAME"
9999 GOTO lineno
```

Saving the program is done by typing GOTO 9998. When the program is reloaded it will auto run keeping the variables safe, you can still break the program and make any alterations, even add new variables to the memory by direct commands (no line numbers) and, provided you didn't use RUN in an absent minded fit, restart the program or resave it as before.

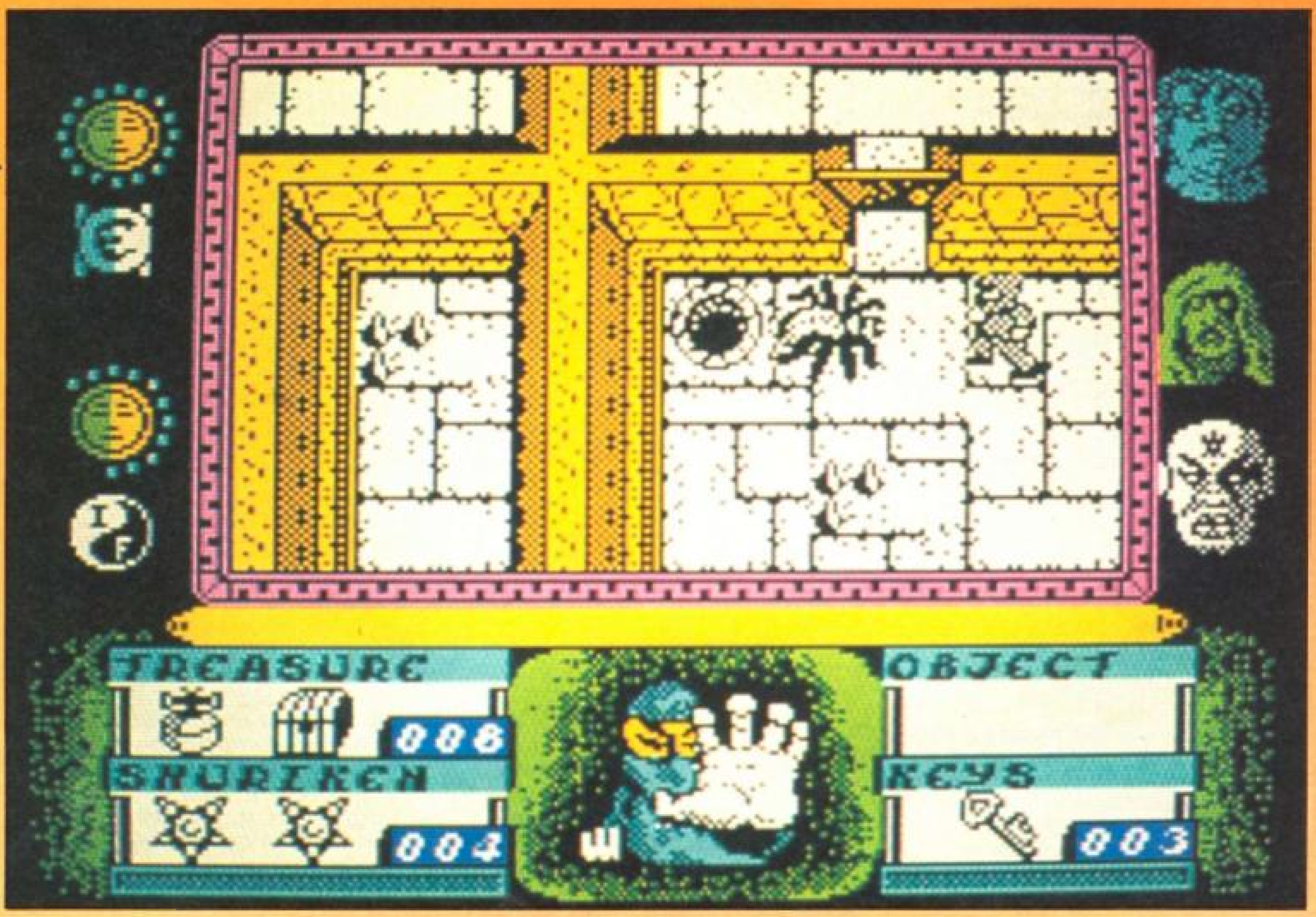
I hope that expert 81'ers will forgive this section of the page, but I met a couple of new users (their dads had donated their old 81s to them and who asked me why I didn't explain some of the accepted practises which we often referred to in our page. Anyway now that is out of the way we'll try and bear in mind that there are newcomers to the machine and try not to talk as if everyone's an expert!

See you next month . . .

PAGE 81



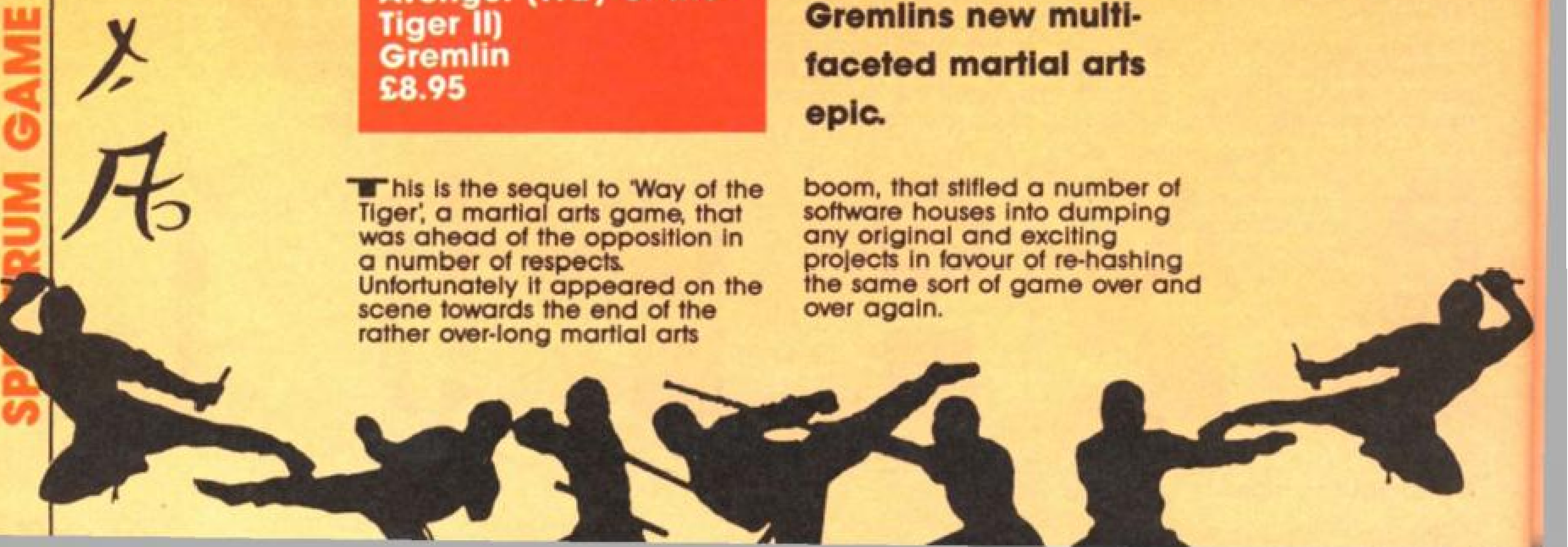
THE WAY OF THE TIGER
Avenger



Avenger (Way of the Tiger II)
Gremlin
£8.95

Ninja revenge is rife in Gremlins new multi-faceted martial arts epic.

This is the sequel to 'Way of the Tiger', a martial arts game, that was ahead of the opposition in a number of respects. Unfortunately it appeared on the scene towards the end of the rather over-long martial arts boom, that stifled a number of software houses into dumping any original and exciting projects in favour of re-hashing the same sort of game over and over again.



It is with great pleasure that I can announce that 'Avenger' is nothing like its prequel, and is a fantastic game that has something for everyone, having elements of adventure, role-playing, (very) fast action, and puzzle/maze/problem solving. The game is based on a format very similar to Gauntlet — the game that has taken over from the martial arts games in setting the trend that all will follow.

Ninja binge

The plot is that having proved your skills as a Ninja in 'Way of the Tiger' you must avenge the murder of your foster father, Naijshi, by the evil Grand Master of Flame — Yaemon — who has stolen the sacred Scrolls of Kettsuin. You swear to the great god Kwon that you will avenge this evil deed and return the scrolls.

The screen is laid out with the action screen in the centre with the three guardians of the keep pictured on the right. Your energy and 'inner peace' levels on the left, and the bottom third of the screen taken up by the status lines. These tell you how many keys you have left, what object you have, how many Ninja throwing stars you have left, and how much treasure you have on your person.

The action screen shows a detailed plan view of your immediate area, with a slight perspective given to everything so that you can see objects tilted slightly — rather than the pure plan view opted for in Gauntlet. The graphics are better than we've come to expect on the Spectrum. Colour has been used creatively and thoughtfully, with none of the livid colour and attribute clashing that mars lesser games. Shading has also been used to great effect, but not over done, so that clarity is maintained.

The Ninja in the centre of the screen is well animated with no jerkiness or sluggishness. The screens scroll with a smoothness that I would have thought impossible on a Spectrum and only occasionally is there a 'flick' from one section of the maze to another, rather than the smooth scroll, as the program unpacks another section of maze.

The mighty Kwon

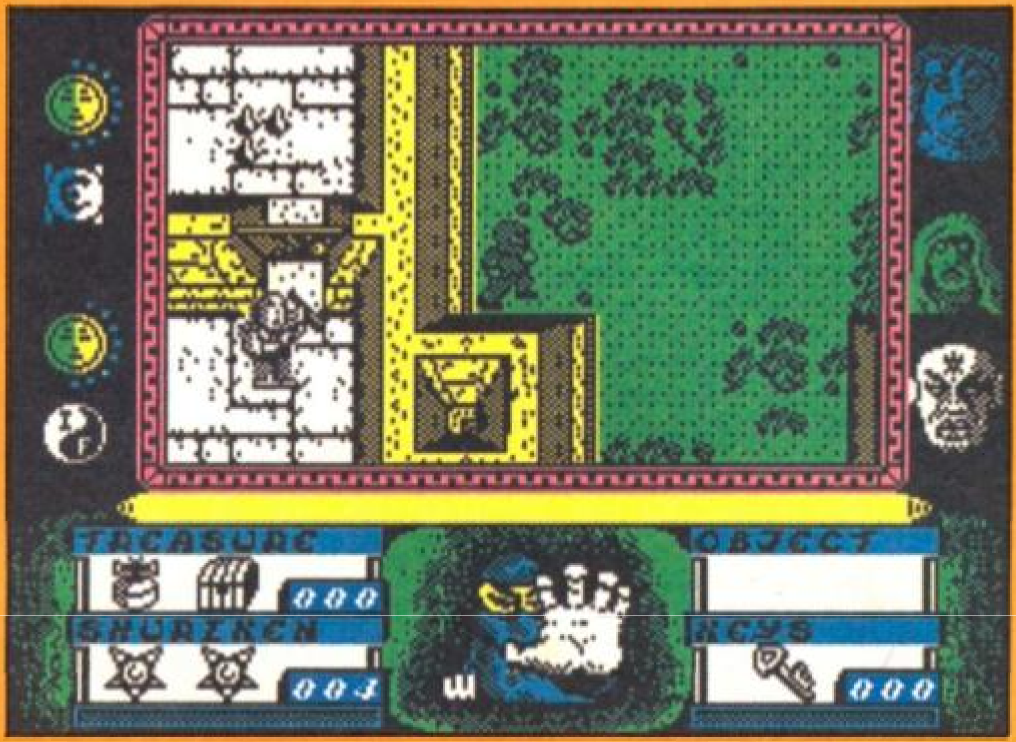
The game is set around performing certain tasks. These are accomplished by collecting various useful objects in a set order as decreed by Kwon (your god). Mapping is essential as collecting keys and opening doors should be done in the right order or you may find yourself unable to progress further because lack of a key.

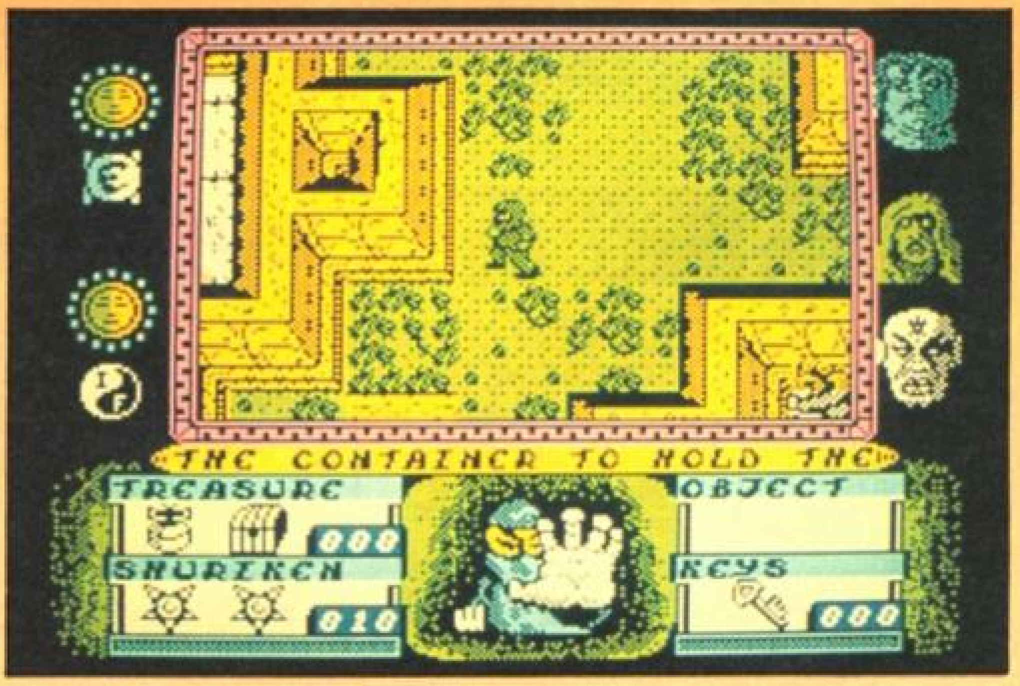
The game is multi-levelled, with gratings in the floor used to descend into the depths of the keep and trap doors to go up (rather like a loft door). In all there are six levels. I have currently seen three of them and the other two show the same kind of thought that went into the level that most people will see. I can only assume that the same was done to the other three levels.

One interesting point is the energy/lives system. When your energy reduces to zero, one point is knocked off your 'inner force'. When this reduces to zero you die. However, praying to Kwon will usually replenish your inner force and revitalise you to carry on his work. However, he is not a patient god and demanding energy too frequently will result in your premature death!

Avenger is excellent, I would recommend it to almost anyone unreservedly.

MONSTER HIT







ZX COMPUTING

FULL CREDIT FACILITIES FOR ACCESS AND BARCLAYCARD.

Lineage: 48p per word. (VAT inclusive) VISA

Semi display: £9.50 per single column centimetre + VAT. Ring for information on series bookings/discounts (minimum 2.5cm).

All advertisements in this section must be prepaid. Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

01 - 437 0699

Send your requirements to:
NICOLA BATY
ASP Ltd., 1 Golden Square,
London W1

UTILITIES

SPECTRUM TAPE and DRIVE TRANSFER UTILITIES

TU8 — ESSENTIAL for ALL Spectrum 48K or Plus owners — our tape utility for backup and drive conversion

* Simple operation for fast loaders — accurate even if over 48K (tone and rate measurer).
* Integrated header reader, manages headerless blocks.
* Deals with clicky loaders — converts COMPLETELY to normal for tape load (plus easy drive transfer with TPM).

Plus many other features — a unique product!

COST only £6.99 on tape OR £7.99 on m/drive cart.

TRANSFER PACK 2 — FOR TAPE TO DRIVE TRANSFER

Package of programs including TU8, other utilities, allows drive transfer of many programs

* Details given on transfer of 40 programs.
* Manages virtually all clicky leaders. Bytes chopper/splitter, header reader etc.

State name of drive system (e.g. m/drive, Opus disc, etc).

ONLY £11.99 on tape OR £12.99 on m/drive cart.

INFORMATION SHEETS £1 each (up to No. 18 available) includes many top 20 hits — SAE for list. OVERSEAS: add £1 Europe, £2 others for each. UPDATE SERVICE: — SAE for details.

PROGRAMS CARRY OUT MONEY BACK GUARANTEE (not updates)

LERM, DEPT ZX, 11 BEACONSFIELD CLOSE, WHITELEY BAY, TYNE AND WEAR NE25 9UW.

SOFTWARE

WD Software

FOR THE QL:

JOSS — **base £13**

Forget that tedious, time-consuming syntax! Just move the cursor and press SPACE for all your file commands. Cursor keys or your joystick allow you to access microdrives (up to 8) and floppy discs (as many as our interface allows) with up to 150 files on each! Scroll and print directories, COPY, DELETE or PRINT any file, select TV or Monitor mode before LOADing or RUNning any program. You only use the keyboard to set the DATE or label a device when FORMATting. Easy to use with Psion and other software. No silly icons to learn - JOSS will TELL you what it's going to do! Programmer's toolkit and mass copying/printing utilities also supplied. Specify microdrive-only, Microperipheral or CST-compatible disc versions.

Ref QL7 — **base £7**

1300 useful QL references with ARCHIVE 2 search/print program Too long for just one cartridge, so if you have RefQL5 just pay 2 and extra media cost to update.

Mdv Extension Cable — **£5.50**

Eight inches long, allows addition of extra microdrives to your QL. Twist it to put their slots facing you.

FOR THE QL: SPECTRUM; BBC; ELECTRON

WD Morse Tutor — **base £4**

Written to teach amateurs, now used by professionals too! Absolute beginner, or stretching your speed to 18 wpm, you won't find anything with more helpful features. What else can offer 100 random sentences as well as all the basics? Disc version unsuitable for BBC B+.

FOR THE SPECTRUM:

WorDfinder (Microdrive/disc only) — **base £8**

For CHEATING at crosswords. Finds m-ss-ng letters, solves, anagrms. 13,000 word vocabulary, so too long to share a cartridge. 10-letter word ending in ATE? No problem.

Tradewind — **base £3**

Sailing/trading strategy game with graphic surprises.

Jersey Quest — **base £3**

Text adventure in time. Background of Jersey folklore from Stone Age to Bergerac.

For export:

QL hardware and software from many sources. Ask for list/quotation.

ORDERING ADD COST OF MEDIUM, POSTAGE £1 OUTSIDE EUROPE.
Mdv or 5.25" floppy = £2 3.5" floppy = £4 Cassette = £0

Payment

By ACCESS/Eurocard/MasterCard or STERLING (UK bank cheques, Eurocheques, drafts or International GIRO). To:

WD Software (ZX), Hilltop, St. Mary, Jersery, C.I. tel: (0534) 81392

QL/SPECTRUM UTILITIES! Wide range of business/practical programs available. SAE/IRC for details (state Micro). S.D. Microsystems (ZX) PO Box 24, Hitchin, Herts.

MAKE RACING PROFITABLE
With our top-class program.
Last season 40% profit.
No racing knowledge needed.
For full details and results write to: **BRIMARDON COMPUTER RACING SERVICE, 48 Pierremont Road, Darlington, DL3 6DH.**

ACCESSORIES

RGB OUTPUT ?
COMPOSITE VIDEO ?
GREEN SCREEN ?
TTL - ANALOGUE ?
COMP SYNC ?
BRIGHT ?

adapt electronics

CONFUSING ISN'T IT?

Talk to the problem solvers. We have the experience and expertise to advise on all aspects of computer display problems. eg:

★ **RGB OUTPUT MODULE** for Spectrum 16 or 48K. Much acclaimed — gives monitor quality pictures — no dot crawl. £36.95

★ **NEW** ★ **BRIGHT BOX** — Allows full use of the bright facility on analogue input monitors and monitor-TV's to produce full colour palette. Only for Spectrum 128. .... £29.95

Add P&P £1 (UK), £2 (EEC), £4 (Others)
Send SAE for our Free Monitor Fact Sheet

20 STARLING CLOSE, BUCKHURST HILL, ESSEX IG9 5TN. Tel: 01-504 2848

SERVICES

STOP PLAYING GAMES

Use your computer to make money. Turn your hobby into a home-based income. Full and part time opportunities to cash in on this tremendous market. High earnings easily possible. Open to any amateur micro user and gamer. Write for free details.

Westlink Promotions; (CG)
108 George Street
Edinburgh EH2 4LH.

ADVERTISING DEADLINES

OUR NEXT COPY DEADLINE IS 19th JANUARY 1987 FOR THE MARCH ISSUE

CLUBS

COMPUTER ENTHUSIASTS!! Would you like to be part of a nationwide - help and be helped - society of enthusiasts with all abilities, machines, interests, (including hardware) catered for? SAE National Computer Club, 212 Dudley Hill Rd, Bradford, BD2 3DF.

NEW ZX81 SOFTWARE Games, utilities, adventures, books, our ZX81 Users Club and much more. Send SAE to: A.C.V., 1 Foxwell Square, Southfields, Northampton NN3 5AT.

NEW ZX81 Hi-Res Game Arcade action with WAR WEB, £3.95. Fun for all the family with POOTER PUZZLER, £2.95. Send S.A.E. for leaflet. Pooter Games, 24 Parsloes Avenue, Dagenham RM9 5NX.

WARNING NOTICE

Advertisements placed in this magazine are to be in strict compliance with our standard conditions (copies of which conditions are available on request) and on the clear understanding that the advertiser warrants that his advertisement(s) does not infringe any copyright or condition of sale of any interested party in the advertised product.

Further, the advertiser indemnifies the proprietors of this magazine in respect of costs, damages, or any other claims brought against them as a result of legal action arising from the publication of the advertisement.

Any breach of these terms or the said conditions may result in prosecution of the advertiser by the proprietors.

UTILITIES

LETTA-HEAD PLUS

Still the most versatile 48k Spectrum utility program to design and print your own business & personal stationery, letterheads, receipts, orders, labels & posters. Create the design on screen, select the required format & print all the copies you need.

- Library of 25 different fonts including foreign alphabets.
- Fast, compiled font editor to modify and create new fonts.
- Any screen portion may be scrolled, inversed or erased.
- Box function encloses any area in a box of any thickness.
- Graphics draw, arc, circle, plot, fill for logo design.
- Load & use any SCREEN$ from another program.
- Print headings in a choice of 18 sizes.
- Design labels up to 4"×2" in size.
- Single key selects cassette label option.
- Choice of different proportionally spaced characters.
- Price £9.00

DUMPY 3.0

All the screen dumps you will ever need for your Spectrum! Unlike other screen dump programs DUMPY is a screen dump GENERATOR; it creates the machine code, relocates it, and saves it ready for you to use in your programs. No need to understand assembler, just follow the menus. The improved version now includes:

- Window dump: definable start & end of both column & line.
- Optional automatic screen area determination.
- Handles the full 24 lines of the display.
- Select from up to 7 different print densities.
- Select from up to 9 widths and 9 heights of output.
- Plain black & white or shaded colour representation.
- Tab to any position on the paper.
- Machine code can be positioned anywhere in RAM.
- Vertical or sideways dumps (for big posters).
- Price £8.50

LIN-O-TYPE

Add style to your written word; print out any wordprocessor file in high density NLQ in a choice of 25 fonts. Turn your Spectrum into an electronic typewriter, superb for addressing envelopes, filling in forms or writing short notes.

- Full on screen WYSIWYG line editor.
- Library of 25 different fonts including foreign alphabets.
- Up to six fonts may be mixed on each line.
- Fast, compiled font editor to modify and create new fonts.
- Single or double width characters and underlining.
- True proportional spacing and right justification.
- Load files from any storage device; mdv, tape, Wafa, disc.
- Print text as found, or edit en route to printer.
- Double density print gives NLQ with most fonts.
- Printing is multi-tasked with keyboard use.
- Price £8.50

ASTRUM+

NEW! The very best Z80 assembler and monitor package for the Spectrum. It is exceedingly versatile, powerful and easy to use and allows unlimited source code to be assembled from drive to drive. Supports microdrives or OPUS DISCOVERY.

- Full screen wordprocessor style editor.
- 256 columns wide for properly documented source code.
- Search & replace. Block copy, delete & move facilities.
- Unlimited size source code in 10k pages.
- Inbuilt macros for easy to read, compact source code.
- Assemble large programs drive to drive - including OPUS!!
- Include source files written with most other assemblers.
- Monitor features up to 9 breakpoints, single stepping etc.
- Slow run facility with 9 breakpoints in RAM or ROM(!).
- Countdown breaks after set number of passes through a loop.
- Library of useful utility programs for file handling.
- Price £13.50 (state microdrive or Opus version).

All Bradway Software programs drive Epson compatible printers (Brother, Star, M-T, Shinwa, Centronics etc.) via Hilderbay, Kempston, Tasman, Morex, Wafadrive, Interface 1, ZX LPrint III and Opus interfaces.

Prices are for cassettes & include P&P within Europe. Please add £1.50 for microdrive cartridge, £2.50 Opus disc, £2.00 worldwide airmail postage. Upgrades from previous versions (where appropriate) £2.00.

Bradway Software (ZX), 33, Conalan Avenue, Sheffield, S17 4PG

REPAIRS

COMPUTER REPAIRS

We are the experts, having serviced Sinclair computers since the introduction of the ZX80.

Don't waste money on estimates – we repair Sinclair computers at price quoted (inclusive parts, labour, postage, VAT, irrespective of fault. No hidden charges.

Repairs guaranteed for 3 months.

Spectrum	£18.75 inc parts
ZX81	£11.50 inc parts
16 KRam	£9.95 inc parts
Microdrive	£15.95 inc parts
Interface 1-11	£18.75 inc parts
also	
BBC	£22.00 + parts
Electron	£19.95 + parts
XK Memory Expansion Kit	£15.95

Computer Retailers please phone for Special Trade Price.

Call or send with cheque or P.O.

T.V. Services of Cambridge Ltd.
French's Road, Cambridge, CB4 3NP
Tel. 0223 311371

SINCLAIR REPAIRS

SPECTRUM/PLUS	£12.00
KEYBOARD	£8.50
INTERFACE/MICRODRIVE	£17.00 each

All prices are fully inclusive of p&p and VAT. Send Cheque with computer only unless power supply suspected. Callers Welcome.

I.T. WESTERN ELECTRONICS
Unit F2A & F3
Avonside Enterprise Park
Newbroughton Road,
Melksham, Wilts
Tele: (0225) 705017

SCOTLAND'S No 1

For home and personal computer repairs
Specially fast Spectrum service!

- *Same day for most faults*
- *1 hour if delivered personally*
- *Open 6 days a week*
- *Free estimates*
- *Upgrades. Membranes and P.S.U.'s*
- *3 mth. warranty on work done*

Also BBC/CBN/ORIC and PERIPHERALS

MICRO-SERV

95 Deerdykes View
Westfield Industrial Area,
Cumbernauld G68 9HN
Scotland
Tel: Cumbernauld (02367) 37110
Trade, schools and club discount given.

SPECTRUM REPAIRS

£14.95 inclusive of labour parts and **p&p.** Fast, reliable service by qualified engs., average repair **24hrs. 3 months guarantee**. For help or advice ring:

H. S. Computer Services, Unit 2, The Orchard, Warton, Preston, Lancashire PR4 1BE. Tel: (0772) 632686.

To advertise your repairs & spares service phone 01-437 0699

SINCLAIR SERVICE CENTRE

- Fully Guaranteed Fast Repair Service
- Approx. 24hr Turnaround
- £15 av cost or send machine with £1.95 (return p&p) for free estimate.
- Personal callers welcome.

QUANTUM
33 CITY ARCADE, COVENTRY CV1 3HX
Tel: (0203) 24632

TERMS

CLASSIFIED ADVERTISING TERMS & CONDITIONS

Our terms for new advertisers (semi-display and lineage) are strictly pro-forma payments until satisfactory reference can be taken up (excluding recognised advertising agencies). Cheques and P.O.'s should be crossed and made payable to ARGUS SPECIALIST PUBLICATIONS LTD. and sent together with the advertisement to:

"The Classified Dept. L/H.
No. 1 Golden Square,
London W1R 3AB.

There are no reimbursements for cancellations. Advertisements arriving too late for a particular issue will be inserted in the following issue unless accompanied by instructions to the contrary.

All advertising sales are subject to Government Regulations concerning VAT. Advertisers are responsible for complying with the various legal requirements in force eg: The Trade Description Act, sex discrimination act & the business advertisements (disclosure) order 1977.

Full Terms & Conditions of Advertising available on request.

Lineage rate: 48p per word (VAT inc.) Minimum **£7.20.**
Semi display: £9.50 per single column centimetre + VAT. Minimum size 2cm. No reimbursements for cancellations.
All ads must be pre-paid.
Write your advert in BLOCK CAPITALS in the grid below, ticking the section you wish it to appear under, INCLUDING YOUR NAME AND ADDRESS IN THE WORD COUNT and send it to: **ZX COMPUTING, ADVERTISEMENT DEPARTMENT, NO: 1 GOLDEN SQUARE, LONDON W1R 3AB.**

☐ REPAIRS ☐ SOFTWARE ☐ UTILITIES ☐ FOR SALE ☐ ACCESSORIES ☐ OTHERWISE PLEASE STATE

CLASSIFIED COUPON

ALL CLASSIFIED ADVERTISEMENTS MUST BE PRE-PAID. THERE ARE NO REIMBURSEMENTS FOR CANCELLATIONS.

I enclose my Cheque/Postal Order for £..........................for...... insertions, made payable to Argus Specialist Publications.
(Delete as necessary)

PLEASE DEBIT MY ACCESS/BARCLAYCARD NO

£ .... FOR .... INSERTIONS EXPIRY DATE ....

Name ....

Address ....

.... POST CODE ....

DAYTIME TEL NO. ....

Signature .... Date ....

IF YOU DO NOT WISH TO CUT YOUR MAGAZINE, PHOTOCOPY THIS FORM

technical graphics

Toni Baker begins a new graphics series on line drawing techniques.

Welcome to a new graphics series. The purpose of this series is to explore the technical possibilities of line drawings. With line drawings we can draw three dimensional objects such as the animated figure which was featured in last month's ZX Computing. In this series we shall keep things nice and simple. We **will** cover the principles of three dimensional drawing, but we **won't** attempt to do complicated things like removing hidden detail or shading in according to the light — that is something for perhaps a future series to cover.

Clipping

The first article is intended to cover the principles of **clipping.** You see, we have to start at the beginning. In order to do any line drawing at all, we must be able to draw straight lines. The DRAW statement in BASIC will draw a straight line successfully provided that the whole of the line will fit on the screen. Now suppose that the line will **not** fit on the screen — for instance PLOT 20,20 followed by DRAW 250,250. If you were to try out this example you'd get the error report "B Integer out of range" and the program would stop running. It would be useful, therefore, to have a routine which would draw only that part of the line which lie within the screen area.

This idea is called "clipping", because a portion of the line at one or both ends will be "clipped" and the line will be shortened. It is of course possible that the entire line will be outside of the screen area, and should be "clipped" away altogether — i.e. nothing at all would be drawn on the screen.

Let's see how this idea of clipping works. Take a look at **Figure 1.** The rectangle represents the screen area. Suppose we want to draw a line from point (P1,Q1) to point (P2,Q2), as shown in the diagram. To clip the line we have to move both end-points closer together, until the line segment will fit on the screen. This is done in four stages. First we find the point (P1',Q1'), which is the point where the line meets the vertical edge of the screen, and then we find the point (P1",Q1"), which is the point where the line meets the horizontal edge of the screen. In the same way we move the point (P2,Q2) first of all to (P2',Q2') and finally to (P2",Q2"). You can see that the line segment from (P1",Q1") to (P2",Q2") lies wholly within the screen and may be drawn normally.

The same principle will work even if the line we wish to clip is not in exactly the position shown in **Figure 1.** Wherever (P1,Q1) lies, we must first calculate (P1',Q1') which will move the point to the nearest vertical edge **only if the point is to the left or to the right of the screen** (otherwise leave it where it is); then we can calculate (P1",Q1") by moving the point if necessary (i.e. if it is above or below the screen) to the nearest horizontal edge.

Special provision must be made for lines which are entirely outside the screen. If both end-points lie to the left of the screen then the whole of the line is also to the left of the screen, and need not be drawn. This is also true if both end-points are to the right of the screen, or if both end-points are above the screen, or if both end-points are below the screen. In each of these cases the line is clipped away completely, and nothing should be drawn at all.

There are other positions of line which lie entirely outside the screen area, and these are more difficult to detect. **Figure 2** shows examples of this kind of line. If the only thing you know about such a line are the co-ordinates of the end-points then it is remarkably difficult for a program to work out whether the

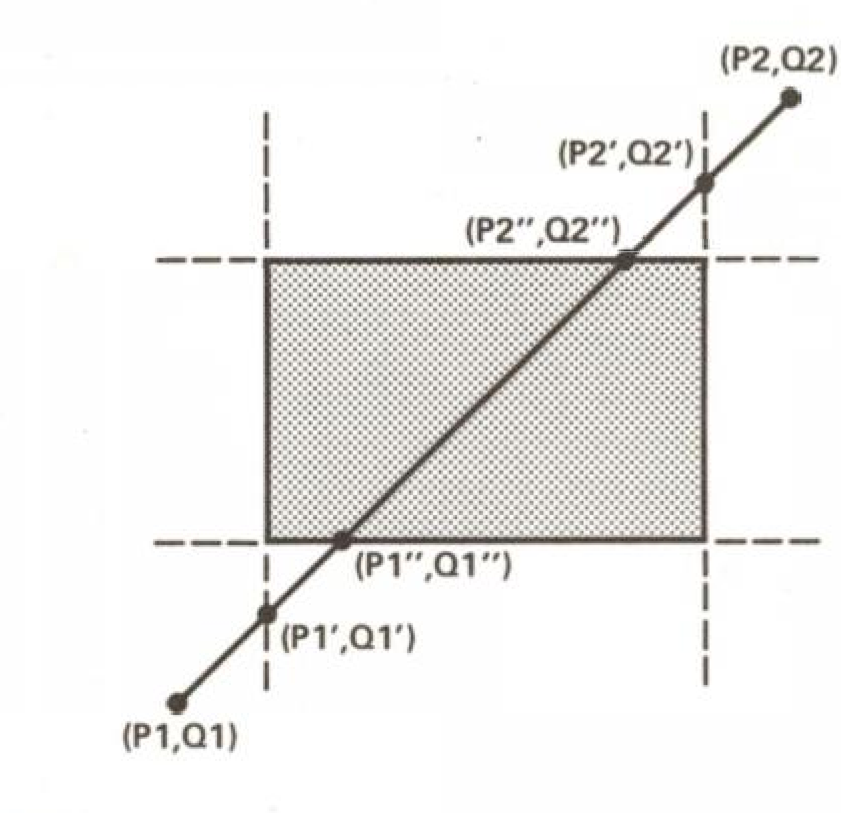


Figure 1

Figure 2

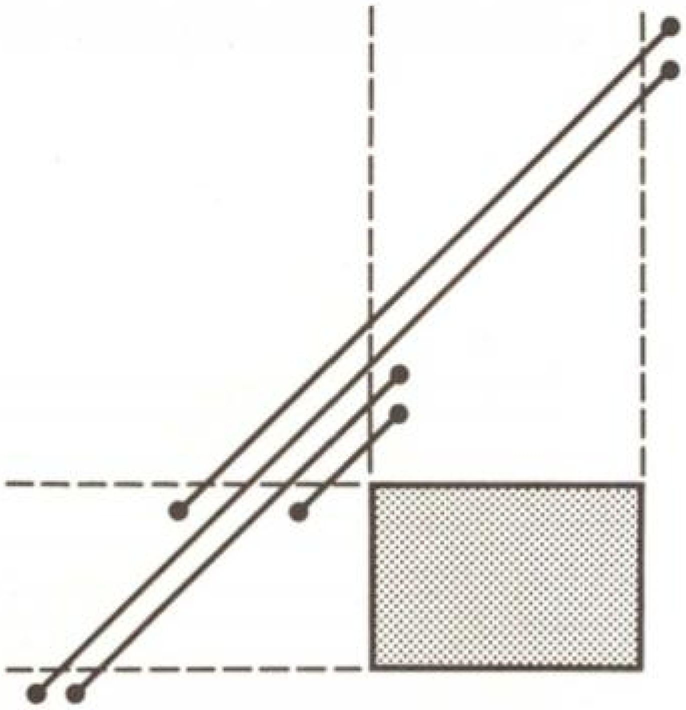
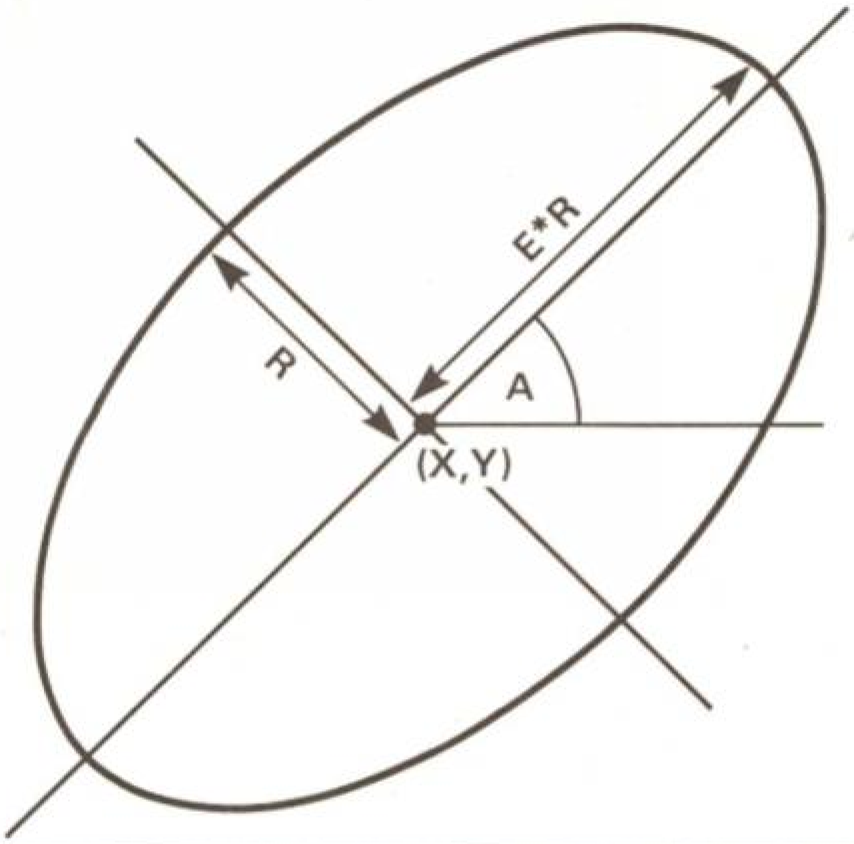


Figure 3



line will or will not intersect the screen. Fortunately, the way round the problem is very simple. All you have to do is to work out P (P1",Q1") and (P2", Q2") using the algorithm above, and if any of the co-ordinates is still outside the screen area then the line should not be drawn.

Getting around

Circles may be drawn by actually drawing a number of very small line segments which give the illusion of a continuous curve. In other words, instead of drawing a circle, we actually draw a many-sided regular polygon. The greater the number of sides, the closer the result is to a circle. The number of sides should ideally be a multiple of four, so that the finished result will be symmetrical. The number you need for any given radius turns out to be roughly PI*SQR(R) where R is the radius — this works well with the resolution possible on the Spectrum.

If we now subject each side of the polygon to the clipping algorithm, so that it will not be drawn if it lies off the screen, or so that only part of it will be drawn if it crosses the screen edge, then you will find that the result is a clipped circle, whereby only that part of the circle which falls on the screen area will be drawn, and the rest will be ignored.

Extending the principle just a little further, we can use the same idea to draw other types of curve than just circles. I have included a routine which will in fact drawn an ellipse — a sort of squashed circle. Since a circle is in fact a special form of ellipse then the same routine will also draw circles.

Although all of the routines are in machine code, I have included some codes which will allow the routines to be used in BASIC. To draw a clipped line whose end-points are (P1,Q1) and (P2,Q2) then you may use the BASIC statement.

RANDOMIZE FN S(P1,Q1,P2,Q2)

To draw a clipped circle whose centre co-ordinates (X,Y) and whose radius is R, then use the BASIC statement.

RANDOMIZE FN C(X,Y,R)

And finally, to draw a clipped ellipse you need five parameters. Use

RANDOMIZE FN E(X,Y,R,E,A)

The parameter list is exactly ▶▶

```
                        ORG 8000h
EF      CONVERT_X       RST 28              S
E8                      recall M8           S,127.5
04                      multiply            127.5*S
31                      duplicate           127.5*S,127.5*S
E0                      recall M0           127.5*S,127.5*S,X1
03                      subtract            127.5*S,127.5*S-X1
E3                      recall M3           127.5*S,127.5*S-X1,Y2
E1                      recall M1           127.5*S,127.5*S-X1,Y2,Y1
03                      subtract            127.5*S,127.5*S-X1,Y2-Y1
04                      multiply            127.5*S,(127.5*S-X1)*(Y2-Y1)
E2                      recall M2           127.5*S,(127.5*S-X1)*(Y2-Y1),X2
E0                      recall M0           127.5*S,(127.5*S-X1)*(Y2-Y1),X2,X1
03                      subtract            127.5*S,(127.5*S-X1)*(Y2-Y1),X2-X1
05                      divide              127.5*S,(127.5*S-X1)*(Y2-Y1)/(X2-X1)
E1                      recall M1           127.5*S,(127.5*S-X1)*(Y2-Y1)/(X2-X1),Y1
0F                      add                 127.5*S,Y1+(127.5*S-X1)*(Y2-Y1)/(X2-X1)
01                      exchange            Y1+(127.5*S-X1)*(Y2-Y1)/(X2-X1),127.5*S
38                      endcalc
C9                      RET                 Return from subroutine.

                        ORG 8013
EF      CONVERT_Y       RST 28              T
E9                      recall M9           T,87.5
04                      multiply            87.5*T
31                      duplicate           87.5*T,87.5*T
E1                      recall M1           87.5*T,87.5*T,Y1
03                      subtract            87.5*T,87.5*T-Y1
E2                      recall M2           87.5*T,87.5*T-Y1,X2
E0                      recall M0           87.5*T,87.5*T-Y1,X2,X1
03                      subtract            87.5*T,87.5*T-Y1,X2-X1
04                      multiply            87.5*T,(87.5*T-Y1)*(X2-X1)
E3                      recall M3           87.5*T,(87.5*T-Y1)*(X2-X1),Y2
E1                      recall M1           87.5*T,(87.5*T-Y1)*(X2-X1),Y2,Y1
03                      subtract            87.5*T,(87.5*T-Y1)*(X2-X1),Y2-Y1
05                      divide              87.5*T,(87.5*T-Y1)*(X2-X1)/(Y2-Y1)
E0                      recall M0           87.5*T,(87.5*T-Y1)*(X2-X1)/(Y2-Y1),X1
0F                      add                 87.5*T,X1+(87.5*T-Y1)*(X2-X1)/(Y2-Y1)
38                      endcalc
C9                      RET                 Return from subroutine.

                        ORG 8025
2A685C  SET_PARAMS      LD HL,(MEM)         HL: points to calculator memories.
E5                      PUSH HL             Stack this address.
3E04                    LD A,04             A:= number of parameters to assign.
3D      SP_LOOP         DEC A               Set zero flag for final pass only.
F5                      PUSH AF             Stack A reg and zero flag.
F6E5                    OR E5
3C                      INC A
47                      LD B,A              B:= E8 (1st and 2nd pass) or E6 (otherwise).
EF                      RST 28              Engage the calculator.
E0                      recall M0           X1
31                      duplicate           X1,X1
2A                      abs                 X1,ABS(X1)
3B                      execute b           X1,ABS(X1),127.5
03                      subtract            X1,ABS(X1)-127.5
37                      gt zero             X1,ABS(X1)>127.5?
0003                    jump true,SP_1      X1          (Jump if ABS(X1)>127.5).
02                      delete
A0                      const zero          0
29      SP_1            sgn                 S1
C4                      store M4            (Memory M4 contains S1).
02                      delete
38                      endcalc
2A685C                  LD HL,(MEM)         HL: points to current memory base.
010500                  LD BC,0005
09                      ADD HL,BC
22685C                  LD (MEM),HL         Memory base now one memory higher up.
F1                      POP AF
20DE                    JR NZ,SP_LOOP       Repeat four times. Note that on 2nd pass
                                            T1 is calculated from Y1, on 3rd pass S2 is
                                            calculated from X2, and on 4th pass T2 is
                                            calculated from Y2, because of the shift of
                                            memory base.
E1                      POP HL
22685C                  LD (MEM),HL         Restore memory base to M0.
C9                      RET                 Return.

                        ORG 8052
EF      CLIP            RST 28              P1,Q1,P2,Q2
34377F                  stk data 87.5       P1,Q1,P2,Q2, 87.5
C9                      store M9            (Memory M9 contains 87.5).
03                      subtract            P1,Q1,P2,Y2
C3                      store M3            (Memory M3 contains Y2).
02                      delete              P1,Q1,P2
34377F                  stk data 127.5      P1,Q1,P2,127.5
C8                      store M8            (Memory M8 contains 127.5)
03                      subtract            P1,Q1,X2
C2                      store M2            (Memory M2 contains X2).
02                      delete              P1,Q1
E9                      recall M9           P1,Q1,87.5
03                      subtract            P1,Y1
C1                      store M1            (Memory M1 contains Y1).
02                      delete              P1
E8                      recall M8           P1,127.5
03                      subtract            X1
C0                      store M0            (Memory M0 contains X1).
02                      delete
38                      endcalc
CD2580                  CALL SET_PARAMS     Assign S1,T1,S2 and T2.
EF                      RST 28
E4                      recall M4           S1
E6                      recall M6           S1,S2
04                      multiply            S1*S2
A1                      const one           S1*S2,1
03                      subtract            S1*S2-1
E5                      recall M5           S1*S2-1,T1
E7                      recall M7           S1*S2-1,T1,T2
04                      multiply            S1*S2-1,T1*T2
A1                      const one           S1*S2-1,T1*T2,1
03                      subtract            S1*S2-1,T1*T2-1
04                      multiply            (S1*S2-1)*(T1*T2-1)
38                      endcalc
CD052D                  CALL FP_TO_A        A:= (S1*S2-1)*(T1*T2-1)
A7                      AND A
C8                      RET Z               Return if entire of line is above, below, or
                                            to one side of, the screen area.
EF                      RST 28
E4                      recall M4           S1
30                      eq zero             S1=0?

0010                    jump true,CLIP_1    (Jump if S1=0).
E4                      recall M4           S1
38                      endcalc             S1
CD0080                  CALL CONVERT_X      Y1',X1'
EF                      RST 28              Y1',X1'
C0                      store M0            (Memory M0 contains X1').
02                      delete              Y1'
C1                      store M1            (Memory M1 contains Y1').
02                      delete
38                      endcalc
CD2580                  CALL SET_PARAMS     (Assign S1' and T1').
EF                      RST 28
E5      CLIP_1          recall M5           T1'
30                      eq zero             T1'=0?
000B                    jump true,CLIP_2    (Jump if T1'=0).
E5                      recall M5           T1'
38                      endcalc             T1'
CD1380                  CALL CONVERT_Y      Y1",X1"
EF                      RST 28              Y1",X1"
C0                      store M0            (Memory M0 contains X1").
02                      delete              Y1"
C1                      store M1            (Memory M1 contains Y1").
02                      delete
E6      CLIP_2          recall M6           S2
30                      eq zero             S2=0?
0010                    jump true,CLIP_3    (Jump if S2=0).
E6                      recall M6           S2
38                      endcalc             S2
CD0080                  CALL CONVERT_X      Y2',X2'
EF                      RST 28              Y2',X2'
C2                      store M2            (Memory M2 contains X2').
02                      delete              Y2'
C3                      store M3            (Memory M3 contains Y2').
02                      delete
38                      endcalc
CD2580                  CALL SET_PARAMS     (Assign S2' and T2').
EF                      RST 28
E7      CLIP_3          recall M7           T2'
30                      eq zero             T2'=0?
000B                    jump true,CLIP_4    (Jump if T2'=0).
E7                      recall M7           T2'


38                      endcalc             T2'
CD1380                  CALL CONVERT_Y      Y2",X2"
EF                      RST 28              Y2",X2"
C2                      store M2            (Memory M2 contains X2").
02                      delete              Y2"
C3                      store M3            (Memory M3 contains Y2").
02                      delete
38      CLIP_4          endcalc
CD2580                  CALL SET_PARAMS     Calculate S1",T1",S2" and T2".
EF                      RST 28
E4                      recall M4           S1"
E5                      recall M5           S1",T1"
07                      or                  S1" OR T1"
E6                      recall M6           S1" OR T1",S2"
07                      or                  S1" OR T1" OR S2"
E7                      recall M7           S1" OR T1" OR S2",T2"
07                      or                  S1" OR T1" OR S2" OR T2"
38                      endcalc             Switch off calculator.
CD052D                  CALL FP_TO_A        A:= S1" OR T1" OR S2" OR T2".
A7                      AND A
C0                      RET NZ              Return if line still outside screen area.
EF                      RST 28              Switch on calculator.
E2                      recall M2           X2"
```

```
E8                  recall M8       X2*,127.5
A2                  const half      X2*,127.5,0.5
0F                  add             X2*,128
C8                  store M8        (Memory M8 contains 128).
0F                  add             X2*+128
27                  int             P2*
C2                  store M2        (Memory M2 contains P2*).
E0                  recall M0       P2*,X1*
E8                  recall M8       P2*,X1*,128
0F                  add             P2*,X1*+128
27                  int             P2*,P1*
C0                  store M0        (Memory M0 contains P1*).
03                  subtract        P2*-P1*
E3                  recall M3       P2*-P1*,Y2*
E9                  recall M9       P2*-P1*,Y2*,87.5
A2                  const half      P2*-P1*,Y2*,87.5,0.5
0F                  add             P2*-P1*,Y2*,88
C9                  store M9        (Memory M9 contains 88).
0F                  add             P2*-P1*,Y2*+88
27                  int             P2*-P1*,Q2*
C3                  store M3        (Memory M3 contains Q2*).
E1                  recall M1       P2*-P1*,Q2*,Y1*
E9                  recall M9       P2*-P1*,Q2*,Y1*,88
0F                  add             P2*-P1*,Q2*,Y1*+88
27                  int             P2*-P1*,Q2*,Q1*
C1                  store M1        (Memory M1 contains Q1*).
03                  subtract        P2*-P1*,Q2*-Q1*
E0                  recall M0       P2*-P1*,Q2*-Q1*,P1*
E1                  recall M1       P2*-P1*,Q2*-Q1*,P1*,Q1*
38                  endcalc
CDDC22              CALL PLOT       PLOT the first point of the line.
C37724              JP LINE_DRAW    DRAW the line segment, and return.
00                  NOP
00                  NOP
00                  NOP             These three bytes unused.
                    ORG 80FD
EF      NXT_POINT   RST 28          Engage the calculator.
EA                  recall MA       E*R
EE                  recall ME       E*R,A1
20                  cos             E*R,COS(A1)
04                  multiply        E*R*COS(A1)
C8                  store M8        (Memory M8 contains E*R*COS(A1)).
EC                  recall MC       E*R*COS(A1),COS(A)
04                  multiply        E*R*COS(A1)*COS(A)
EB                  recall MB       E*R*COS(A1)*COS(A),R
EE                  recall ME       E*R*COS(A1)*COS(A),R,A1
1F                  sin             E*R*COS(A1)*COS(A),R,SIN(A1)
04                  multiply        E*R*COS(A1)*COS(A),R*SIN(A1)
C7                  store M7        (Memory M7 contains R*SIN(A1)).
ED                  recall MD       E*R*COS(A1)*COS(A),R*SIN(A1),SIN(A)
04                  multiply        E*R*COS(A1)*COS(A),R*SIN(A1)*SIN(A)
03                  subtract        E*R*COS(A1)*COS(A)-R*SIN(A1)*SIN(A)
F2                  recall M12      E*R*COS(A1)*COS(A)-R*SIN(A1)*SIN(A),X
0F                  add             X+E*R*COS(A1)*COS(A)-R*SIN(A1)*SIN(A)
D0                  store M10       (Memory M10 contains P ).
02                  delete
E8                  recall M8       E*R*COS(A1)
ED                  recall MD       E*R*COS(A1),SIN(A)
04                  multiply        E*R*COS(A1)*SIN(A)
E7                  recall M7       E*R*COS(A1)*SIN(A),R*SIN(A1)
EC                  recall MC       E*R*COS(A1)*SIN(A),R*SIN(A1),COS(A)
04                  multiply        E*R*COS(A1)*SIN(A),R*SIN(A1)*COS(A)
0F                  add             E*R*COS(A1)*SIN(A)+R*SIN(A1)*COS(A)
F3                  recall M13      E*R*COS(A1)*SIN(A)+R*SIN(A1)*COS(A),Y
0F                  add             Y+E*R*COS(A1)*SIN(A)+R*SIN(A1)*COS(A)
D1                  store M11       (Memory M11 contains Q ).
02                  delete
38                  endcalc
C9                  RET             Return from subroutine.

                    ORG 811E
EF      ELLIPSE     RST 28          X,Y,R,E,A
31                  duplicate       X,Y,R,E,A,A
20                  cos             X,Y,R,E,A,COS(A)
CC                  store MC        (Memory MC contains COS(A)).
02                  delete          X,Y,R,E,A
1F                  sin             X,Y,R,E,SIN(A)
CD                  store MD        (Memory MD contains SIN(A)).
02                  delete          X,Y,R,E
01                  exchange        X,Y,E,R
CB                  store MB        (Memory MB contains R).
04                  multiply        X,Y,E*R
CA                  store MA        (Memory MA contains E*R).
02                  delete          X,Y
D3                  store M13       (Memory M13 contains Y).
02                  delete          X
D2                  store M12       (Memory M12 contains X).
02                  delete
A0                  const zero      0
CE                  store ME        (Memory ME contains zero).
02                  delete
EB                  recall MB       R
2A                  abs             ABS(R)
31                  duplicate       ABS(R),ABS(R)
EA                  recall MA       ABS(R),ABS(R),E*R
2A                  abs             ABS(R),ABS(R),ABS(E*R)
03                  subtract        ABS(R),ABS(R)-ABS(E*R)
37                  gt zero         ABS(R),ABS(R)>ABS(E*R)?
0004                jump true,ELL_1 ABS(R)    (Jump if ABS(R)>ABS(E*R)).
02                  delete
EA                  recall MA       E*R
2A                  abs             ABS(E*R)
        ELL_1                       M         (=ABS(R) or ABS(E*R), whichever
                                              is the larger).
28                  sqr             SQR(M)

A3                  const pi/2      SQR(M),PI/2
04                  multiply        SQR(M)*PI/2
38                  endcalc         SQR(M)*PI/2
34                  INC (HL)        SQR(M)*PI
CDD52D              CALL FP_TO_A    A:= SQR(M)*PI to nearest integer.
3806                JR C,ELL_FC     Jump if SQR(M)*PI>255d.
E6FC                AND FC
C604                ADD A,04        A:= number of arcs needed to draw ellipse.
3002                JR NC,ELL_DRAW  Jump if<256d.
3EFC    ELL_FC      LD A,FC         Use 252d arcs.
F5      ELL_DRAW    PUSH AF         Stack number of arcs needed.
CD282D              CALL STACK_A    Push onto calculator stack.
EF                  RST 28          ARCS
A3                  const pi/2      ARCS,PI/2
38                  endcalc         ARCS,PI/2
3683                LD (HL),83      ARCS,2*PI
EF                  RST 28          ARCS,2*PI
01                  exchange        2*PI,ARCS
05                  divide          2*PI/ARCS
CF                  store MF        (Memory MF contains the amount by which the
                                    angle must be incremented at each pass).
02                  delete
38                  endcalc
CDFD80              CALL NXT_POINT  (Memory M10 contains P1, and M11 contains Q1).
C1                  POP BC          B:= ARCS.
C5      ELL_LOOP    PUSH BC
EF                  RST 28
EE                  recall ME       A1
EF                  recall MF       A1,INCR
0F                  add             A1+INCR
CE                  store ME        (Memory ME contains updated angle A1).
02                  delete
F0                  recall M10      P1
F1                  recall M11      P1,Q1
38                  endcalc         P1,Q1
CDFD80              CALL NXT_POINT  (Memory M10 contains P2, and M11 contains Q2).
EF                  RST 28          P1,Q1
F0                  recall M10      P1,Q1,P2
F1                  recall M11      P1,Q1,P2,Q2
38                  endcalc         P1,Q1,P2,Q2
CD5280              CALL CLIP       Draw the line segment.

C1                  POP BC          B:= Arc count.
10E9                DJNZ ELL_LOOP   Loop back to draw entire of ellipse.
C9                  RET             Return.

                    ORG 817B
013200  FN_SEGMENT  LD BC,0032      BC:= 5 * 10d.
CDA681              CALL SET_UP     Create ten calculator memories, and push
                                    P1,Q1,P2,Q2 onto calculator stack.
CD5280              CALL CLIP       Draw the clipped line segment.
1815                JR FN_EXIT      Jump to exit.
016400  FN_CIRCLE   LD BC,0064      BC:= 5 * 20d.
CDA681              CALL SET_UP     Create 20d mems & push X,Y,R on calc stack.
EF                  RST 28          X,Y,R
A1                  const 1         X,Y,R,1
A0                  const 0         X,Y,R,1,0
38                  endcalc
1806                JR FN_ELL_2     Jump to draw the clipped circle.
016400  FN_ELLIPSE  LD BC,0064      BC:= 5 * 20d.
CDA681              CALL SET_UP     Create 20d mems & push X,Y,R,E,A on calc stk.
CD1E81  FN_ELL_2    CALL ELLIPSE    Draw the ellipse (or circle).
21925C  FN_EXIT     LD HL,MEMBOT
22685C              LD (MEM),HL     Restore calculator memories to normal.
215827              LD HL,2758
D9                  EXX             HL':= 2758 to prevent system crash.
C9                  RET             Return to BASIC.

                    ORG 81A6
F7      SET_UP      RST 30          Create space for calculator memories.
ED53685C            LD (MEM),DE     Move calculator memories accordingly.
2A0B5C              LD HL,(DEFADD)  HL: pts to 1st user-defined FN argument record.
23      SU_LOOP     INC HL
23                  INC HL          HL: points to next FN argument.
CDB433              CALL STACK_NUM  Push this argument onto the calc stack.
7E                  LD A,(HL)
23                  INC HL
FE2C                CP 2C
28F5                JR Z,SU_LOOP    Loop back if there are more arguments.
C9                  RET             Return.
```

the same as that on the QL. X and Y are the co-ordinates of the centre of the ellipse. R is the smaller radius, and E (which I've assumed to be greater than one) is the ratio of the larger to the smaller radius. A is the angle (in radians) which the larger radius makes with the horizontal. **Figure 3** shows how all these parameters fit together to define a full elipse. Note that if E is less than one then for "larger" read "smaller" and vice versa in the above description. Note also that if E equals one then the "ellipse" will actually be a circle.

The program makes use of twenty calculator memories (ten for the clipping routine, and ten more which deal with ellipse drawing). Since the calculator only has six memories built in then it is necessary to create the extra memory in the workspace using the RST 30 instruction, and then point the system variable (MEM) to this memory. (MEM) should be returned to its normal value of 5C92 at the end of the routine, and this is done at the label FN_EXIT near the end of the machine code program. The actual usage of these memories is detailed in **Figure 4.**

Finally, there is included a BASIC program **(Figure 5)** which demonstrates the clipping routines by drawing a rather nice pattern out of lines, circles and ellipses, which are too large to fit on the screen. Give it a run and see what you think.

In machine code, the main clipping routine is the routine labelled CLIP at address 8052, which requires P1,Q1,P2,Q2 at the top of the calculator stack, in that order. The routine ELLIPSE at address 811E will draw an ellipse. It requires the parameters X,Y,R,E,A, in that order, at the top of the calculator stack. The entry point from BASIC will be either FN_SEGMENT (address 817B), FN_CIRCLE (address 8186) or FN_ELLIPSE (address 8192), depending upon whether FN S, FN C, or FN E were used.

In the next article in this series, we'll start taking a look at 3D, beginning with Isometric and other simple kinds of projection (don't worry — it's easy). See you then.

P.S. Thanks to M.P. Computers for the speedy repair of my microdrive unit.

M0	X1	X coordinate of start of line - 127.5
M1	Y1	Y coordinate of start of line - 87.5
M2	X2	X coordinate of end of line - 127.5
M3	Y2	Y coordinate of end of line - 87.5
M4	S1	-1,0 or 1, if X1 is left of, on, or right of screen respectively.
M5	T1	-1,0 or 1, if Y1 is below, on, or above screen respectively.
M6	S2	-1,0 or 1, if X2 is left of, on, or right of screen respectively.
M7	T2	-1,0 or 1, if Y2 is below, on, or above screen respectively.
M8	127.5	Half the width of the screen.
M9	87.5	Half the height of the screen.
MA	E*R	Half the major axis of ellipse.
MB	R	Half the minor axis of ellipse.
MC	COS(A)	)- Inclination of major axis to horizontal.
MD	SIN(A)	)
ME	A1	Angle subtended "so far".
MF	INCR	Amount by which A1 is incremented on each pass.
M10	P	X coordinate of point on ellipse.
M11	Q	Y coordinate of point on ellipse.
M12	X	X coordinate of centre of ellipse.
M13	Y	Y coordinate of centre of ellipse.

Figure 4

```
 10 FOR I = 0 TO PI STEP PI/20
 20 RANDOMIZE FN E(128,88,60,2,I)
 30 NEXT I
 40 FOR I = 60 to 120 STEP 20
 50 FOR J = 0 TO 4
 60 LET A1 = (2*J/5+.5)*PI
 70 LET A2 = (2*(J+2)/5+.5)*PI
 80 RANDOMIZE FN S(128+I*COS A1,88+I*SIN A1,128+I*COS A2,88+I*SIN A2)
 90 NEXT J
100 RANDOMIZE FN C(128,88,I)
110 NEXT I
120 DEF FN S(A,B,C,D) = USR 33147
130 DEF FN C(X,Y,R) = USR 33158
140 DEF FN E(X,Y,R,E,A) = USR 33170
```

Figure 5

SCALEXTRIC®



This split-level racing game incorporates all the excitement of the race track, in fact many race tracks, as it features an icon-driven track designer that allows you to construct your own tracks as well as select scaled Formula 1 championship circuits.

One or two people can play and have their own 3D view. Some ingenious programming has made the feel of controlling the cars very realistic, and you can even bump and push your opponent's car if you want to play dirty!

Available for **Commodore 64** at **£9.95**

NOW AVAILABLE FOR SPECTRUM 48/128 AT **£9.95**

and coming soon for AMSTRAD CPC computers!

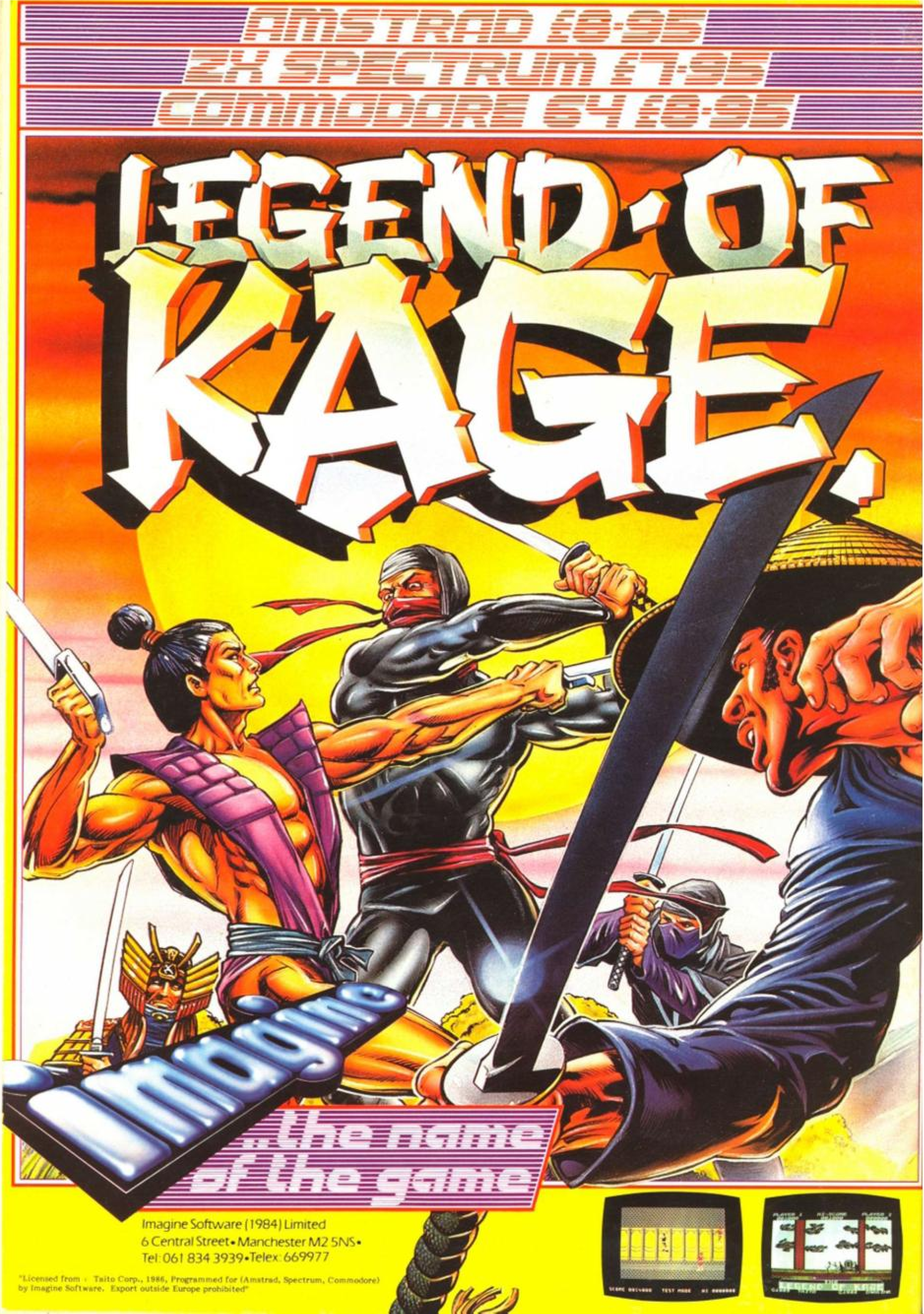
Please send me a copy of **SCALEXTRIC** at **£9.95**
COMMODORE 64/128 ☐ SPECTRUM 48/128 ☐

Name ______________________

Address ______________________

Please make cheques and postal order payable to **Virgin Games Ltd.**, and post to **Virgin Games, 2/4 Vernon Yard, Portobello Road, London W11 2DX.** *Please do not post cash.*

Virgin
Leisure Genius



AMSTRAD £8·95
ZX SPECTRUM £7·95
COMMODORE 64 £8·95
LEGEND·OF
KAGE.
Imagine
..the name
of the game
Imagine Software (1984) Limited
6 Central Street • Manchester M2 5NS •
Tel: 061 834 3939 • Telex: 669977
"Licensed from : Taito Corp., 1986, Programmed for (Amstrad, Spectrum, Commodore) by Imagine Software. Export outside Europe prohibited"
LEGEND OF KAGE